**PCT** WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30

A2

(11) International Publication Number: WO 99/23584

(43) International Publication Date: 14 May 1999 (14.05.99)

(21) International Application Number: PCT/US98/23193

(22) International Filing Date: 2 November 1998 (02.11.98)

(30) Priority Data:

| | | |
|---|---|---|
| 08/961,714 | 31 October 1997 (31.10.97) | US |
| 08/962,117 | 31 October 1997 (31.10.97) | US |

(71) Applicant *(for all designated States except US)*: IOTA INDUSTRIES LTD. [IL/IL]; Rechov Habarzel 31, 69710 Tel Aviv (IL).

(71) Applicant *(for TJ only)*: FRIEDMAN, Mark, M. [US/IL]; Alharizi 1, 43406 Raanana (IL).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: STERN, Yonatan, Pesach [IL/IL]; Beit Tzuri Street 7/4, Tel Aviv (IL).

(74) Common Representative: FRIEDMAN, Mark, M.; c/o Castorina, Anthony, Suite 207, 2001 Jefferson Davis Highway, Arlington, VA 22202 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

**Published**

*Without international search report and to be republished upon receipt of that report.*

(54) Title: INFORMATION COMPONENT MANAGEMENT SYSTEM

(57) Abstract

A management system for managing information, in which information is first captured from an original source having an original format, and then converted into an information component. The information component can be stored in, and retrieved from, a database. Upon being retrieved from a database, the original information can be extracted from the information component and displayed in a substantially similar format as the original information.

*FOR THE PURPOSES OF INFORMATION ONLY*

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| AL | Albania | ES | Spain | LS | Lesotho | SI | Slovenia |
| AM | Armenia | FI | Finland | LT | Lithuania | SK | Slovakia |
| AT | Austria | FR | France | LU | Luxembourg | SN | Senegal |
| AU | Australia | GA | Gabon | LV | Latvia | SZ | Swaziland |
| AZ | Azerbaijan | GB | United Kingdom | MC | Monaco | TD | Chad |
| BA | Bosnia and Herzegovina | GE | Georgia | MD | Republic of Moldova | TG | Togo |
| BB | Barbados | GH | Ghana | MG | Madagascar | TJ | Tajikistan |
| BE | Belgium | GN | Guinea | MK | The former Yugoslav Republic of Macedonia | TM | Turkmenistan |
| BF | Burkina Faso | GR | Greece | | | TR | Turkey |
| BG | Bulgaria | HU | Hungary | ML | Mali | TT | Trinidad and Tobago |
| BJ | Benin | IE | Ireland | MN | Mongolia | UA | Ukraine |
| BR | Brazil | IL | Israel | MR | Mauritania | UG | Uganda |
| BY | Belarus | IS | Iceland | MW | Malawi | US | United States of America |
| CA | Canada | IT | Italy | MX | Mexico | UZ | Uzbekistan |
| CF | Central African Republic | JP | Japan | NE | Niger | VN | Viet Nam |
| CG | Congo | KE | Kenya | NL | Netherlands | YU | Yugoslavia |
| CH | Switzerland | KG | Kyrgyzstan | NO | Norway | ZW | Zimbabwe |
| CI | Côte d'Ivoire | KP | Democratic People's Republic of Korea | NZ | New Zealand | | |
| CM | Cameroon | | | PL | Poland | | |
| CN | China | KR | Republic of Korea | PT | Portugal | | |
| CU | Cuba | KZ | Kazakstan | RO | Romania | | |
| CZ | Czech Republic | LC | Saint Lucia | RU | Russian Federation | | |
| DE | Germany | LI | Liechtenstein | SD | Sudan | | |
| DK | Denmark | LK | Sri Lanka | SE | Sweden | | |
| EE | Estonia | LR | Liberia | SG | Singapore | | |

# INFORMATION COMPONENT MANAGEMENT SYSTEM

## FIELD AND BACKGROUND OF THE INVENTION

The present invention relates to an information component management system. Specifically, the system of the present invention enables documents, images and other types of information to be packaged within an active information component object, which can then be stored, retrieved and manipulated according to content rather than according to form.

Both the amount and format of available information is increasing at a geometric rate. Individuals today face a plethora of choices, both of the type of information which can be obtained, and the method by which the information is obtained. For example, in addition to the traditional print media such as newspapers and magazines, a good deal of news information is available electronically, via the World Wide Web (WWW), through electronic computer mail, by dedicated electronic news services, through a facsimile machine or even on television. All of this information can be obtained relatively easily, yet finding particularly useful information is increasingly difficult if not impossible.

The many different information formats are themselves a source of increasing complexity for information management. Such management includes storing, searching and retrieving available information to find that small fraction which is useful to the user. For example, a particular news item might be available in a paper document, as a picture, from a video stream such as television broadcast, through a voice medium such as radio, or electronically on the World Wide Web. As its name implies, a "document" management system is still tied to the underlying characteristics of a "document".

Documents can be defined as a collection of ideas and information, which are organized within a certain structure. The ideas and information may be logically linked according to various relationships, but as a whole should follow a common theme. The collection itself is expressed as a combination of text and graphic items. There are three main types of information in a document: ideas, data and structure. Ideas can be expressed with words or graphics. Data can be in the form of numbers, symbols, graphics or even sounds. The final element, structure, is an important element of a document, yet it is often overlooked as a separate entity. The structure of a document is the way in which the data and ideas are organized within the document, thereby providing additional significance to these data and ideas.

Current document management systems typically fall into one of two categories. The first category is a document management system. This system was originally designed to

enable searches for information according to specific keywords within defined database fields. Unfortunately, this underlying system design has many disadvantages. For example, the types of searches are limited by the structure of the database itself. Furthermore, information must be extracted from the document and entered into the database manually, which is time consuming, expensive and prone to human error. Thus, structured management systems have significant drawbacks for document management.

The alternative category, non-structured text retrieval systems, solves certain problems but also creates new difficulties. These systems enable automatic indexing of information, without the need for human intervention. However, in non-structured retrieval systems, only the free text of the document is automatically indexed. Therefore, only free text from the document can be searched. Although free text is an important component of a document, such a system loses the other types of available information. Furthermore, the context of ideas or concepts within a document is largely lost by the automatic indexing procedure, leaving the user with a collection of disconnected textual segments or documents which are divorced from the general theme expressed by the entire document. Thus, the user must often read an entire document or a collection of search results in order to find the desired information.

Therefore, there is an unmet need for, and it would be highly useful to have, an information component retrieval system which stores, manages and retrieves concepts and ideas rather than static documents or document portions.

## SUMMARY OF THE INVENTION

The information component management system of the present invention enables documents, images and other types of information to be packaged within an active information component object, which can then be stored, retrieved and manipulated according to content rather than according to form. The information component includes concepts or ideas, data and structure as separate but related entities. Information components are linked to each other according to a particular relationship, which may be either parallel or hierarchical.

According to the present invention, there is provided an information component system for storing an original document, comprising: (a) at least one information component for storing information from the original document, the at least one information component featuring at least one information component primitive; (b) an information component identifier for classifying the at least one information component according to at least one

information component class; and (c) at least one property of the at least one information component.

According to another embodiment of the present invention, there is provided a system for displaying a native file format document, the document including text and having a native file format and a native document appearance, the native file format including at least one instruction for displaying the text of the native file format document, the system comprising: (a) a Web browser for displaying the native file format document according to the native document appearance; and (b) a HTML rendering engine for obtaining information regarding the native document appearance of the native file format document, for translating the information into a raster file having a raster format displayable by the Web browser, and for giving the translated information to the Web browser, such that the Web browser is able to display the native file format document.

According to still another embodiment of the present invention, there is provided a method for managing information, comprising the steps of: (a) capturing the information in an electronic format; (b) converting the captured information into an information component, the information component featuring: (i) a pointer to a storage location of the captured information; (ii) at least one method for manipulating the captured information; and (iii) at least one property of the captured information; (c) storing the information component; and (d) displaying the information component such that the captured information appears in substantially the original format.

According to yet another embodiment of the present invention, there is provided an information component comprising a software object, the software object including: (a) a pointer to a storage location of the stored original information;(b) at least one method for manipulating the stored original information; and (c) at least one property of the stored original information.

According to still another embodiment of the present invention, there is provided a server for serving stored information to a client Web browser, the server comprising: (a) a database for storing the stored information; and (b) an image processor for accessing the stored information from the database and transforming the stored information into a Searchable Image Format (SIF) file, the SIF file being accessed by the client Web browser, such that the stored information is displayed by the client Web browser.

Hereinafter, the term "computing platform" refers to a particular computer hardware system or to a particular software operating system. Examples of such hardware systems include, but are not limited to, personal computers (PC), Mackintosh™ computers,

mainframes, minicomputers and workstations. Examples of such software operating systems include, but are not limited to, UNIX, VMS, Linux, MacOS™, DOS, one of the Windows™ operating systems by Microsoft Inc. (Seattle, Washington, USA), including Windows NT™, Windows 3.x™ (in which "x" is a version number, such as "Windows 3.1™") and Windows95™. Hereinafter, the term "software object" includes any software application capable of substantially independent execution by an operating system. For the present invention, a software application, whether a software object or substantially any other type of software application, could be written in substantially suitable programming language, which could easily be selected by one of ordinary skill in the art. The programming language chosen should be compatible with the computing platform according to which the software application is executed. Examples of suitable programming languages include, but are not limited to, C, C++ and Java.

Hereinafter, the term "Web browser" refers to any software program which can be used to view a document written at least partially with at least one instruction taken from HTML (HyperText Mark-up Language) or VRML (Virtual Reality Modeling Language), or any other equivalent computer document language, hereinafter collectively and generally referred to as "document mark-up language". Examples of Web browsers include, but are not limited to, Mosaic™, Netscape Navigator™ and Microsoft™ Internet Explorer™. Hereinafter, the term "raster format" refers to any image format supported by Web browsers including, but not limited to, GIF (Graphics Interchange Format), JPEG (Joint Photographics Expert Group) and PNG (Portable Network Graphics).

## BRIEF DESCRIPTION OF THE DRAWINGS

The invention is herein described, by way of example only, with reference to the accompanying drawings, wherein:

FIGS. 1A-1C are schematic block diagrams of various exemplary information components and classes;

FIGS. 2A and 2B are schematic block diagrams of the general architecture of an exemplary system of the present invention;

FIGS. 3A and 3B are schematic block diagrams of a preferred embodiment of the IC Identifier of the present invention;

FIG. 4 is a schematic block diagram of an exemplary IC component generator of the present invention;

FIG. 5 is a schematic block diagram of a preferred embodiment of the IC Server of the present invention;

FIG. 6 is a schematic block diagram of a preferred embodiment of the IC Client of the present invention;

FIG. 7 is a schematic block diagram of a preferred embodiment of the system of the present invention as implemented in XML;

FIG. 8 is a schematic block diagram of dynamic link management according to the present invention;

FIG. 9 is a schematic block diagram of DTD normalization according to the present invention;

FIG. 10 is a schematic block diagram of an exemplary system for HTML rendering according to the present invention;

FIG. 11 shows an exemplary output of the system of Figure 10;

FIG. 12 shows an exemplary embodiment of the IC Server of the present invention as implemented with Java Bean objects;

FIG. 13 shows an exemplary embodiment of the IC Client of the present invention as implemented with Java Bean objects.

## GENERAL DESCRIPTION OF THE INVENTION

The information component management system of the present invention enables documents, images and other types of structured or non-structured information to be analyzed. The underlying structure of the information is determined, and the structure is exposed to a database.

The information is then packaged within an active information component object, which can then be stored, retrieved and manipulated according to content rather than according to form. The information component includes concepts or ideas, data and structure as separate but related entities. Information components are linked to each other according to a particular relationship, which may be either parallel or hierarchical.

For example, an image of a face of a person is an information component which may in turn be a portion of a larger object, such as a group photo, which may in turn be a portion of an article. The image of the face, the group photo and the article are all individual information components which are linked according to a hierarchical structure. Each information component inherits the features of all associated information components which are higher in the hierarchical structure, and in turn contributes to the pool of features

characterizing associated information components which are lower in the hierarchical structure. Thus, information components have both content related to the actual stored information, and content related to the features of associated, higher level components.

The actual stored information from an information component is displayed in substantially the same format as the original source format, so as to maintain the original appearance as much as possible. The displayed information maintains substantially the same fonts, graphics and structure, so that a newspaper page is displayed as a substantially exact reproduction of the page as it originally appeared in newsprint, for example. Thus, the system of the present invention has a clear advantage over prior art document management systems, which usually display retrieved information only as text. Even if graphic images are also displayed, the structure of the entire document, and the visual relationship between the text and the images, is not maintained by these prior art systems.

The information component management system of the present invention is able to search for, and retrieve, information based upon all characteristics of the information component, including graphic images, text and structural relationships. Results are presented as intuitive, visually explicit objects which are easy to examine, manipulate and navigate through. Furthermore, the search results are presented according to the ranked relevance to the desired search strategy, in which the rank is determined with both the full content and the complete characteristics of the information component.

Thus, the system of the present invention includes two basic principles: object oriented management and visual information retrieval. Both principles will be explained in greater detail below, in the Description of the Preferred Embodiments. Briefly, the information components are managed as objects which belong to an information class. Different information classes are linked according to the logical relationship between the components in each class. Overall, the classes are placed within a hierarchical structure, in which each child class inherits the properties of the parent class. Each information class defines the properties and operations of a set of information component.

As noted previously, each information component is a representation of information, combining structured and non-structured data. As an object, the information component also features methods for accessing and manipulating the information, including the data interface and any data operations. Because the methods of the information component are exposed to the general computational environment, the component either can be displayed, or can display itself, on any type of computing platform or operating system. Thus, the information

component is both compatible across different computing platforms and has an open, easily accessible interface.

In order to prepare such an information component, several procedures must be performed. First, the information must be identified. Next, the information must be classified and the actual information component must be created. The relationship between the new information component and other information component(s) must be identified. Finally, the behavior of the completed information component is determined according to the functionality of the attributes or features which accrue to that component after classification and identification of relationships.

Once prepared, the information component can be searched and retrieved through visual information search and retrieval. Briefly, the search can be performed according to keyword. visual example and graphic attributes. Visual examples include images or graphic objects which are compared to graphic information stored in the database, just as a keyword search involves the comparison of keywords to text stored in the database. Graphic attributes include font size, font attribute and relative positioning of information within a document. These attributes can also be used as search parameters. Thus, the search is not limited to a simple keyword comparison of stored textual information.

Information which is retrieved as a result of the search is then presented in a substantially similar or even identical format as the original source format. Furthermore, the relevance ranking of the retrieved information is preferably determined both by the number and density of required keywords which appear in the information component, if any, but also is preferably calculated according to the desired visual attributes and relationships to other information components. Even more preferably, as described in more detail below, the system of the present invention includes a mechanism for learning the preferences and profile of an individual user, which can then also be used to calculate the relevance ranking of the retrieved information.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention is of an information component management system, in which information is packaged as an information component, including textual data, images and structure. Information components are related to each other according to a hierarchical organization, in which characteristics of components which are higher in the hierarchy accrue to those components which are lower in the hierarchy. The information components can be searched and retrieved according to all attributes of the actual information, as well as the

characteristics of the component and relationships between components. Thus, the information component management system of the present invention is not limited to simple storage, searches and retrieval of textual data only, but instead preserves all aspects of the original source of information.

The principles and operation of the information component system according to the present invention may be better understood with reference to the drawings and the accompanying description. It should be noted that the following description will make reference to the Java computer programming language and to related software architecture, it being understood that this is for the sake of clarity only and is not meant to be limiting in any way.

The detailed description of the system of the present invention will be divided into four chapters. The first chapter describes various background art technologies which are the preferred support technologies for the system of the present invention. These technologies are described as "background art" because they are not fulfilling the same functions as the system of the present invention, but instead are merely enabling these functions. These technologies are given as examples only and are not intended to be limiting in any way.

The second chapter provides a brief overall view of the entire system according to the present invention. The third chapter describes an exemplary implementation of the present invention with objects in an XML environment, and the fourth chapter describes an exemplary implementation of the present invention with Java Bean objects in a CORBA environment.

## Chapter I: Background Art Technologies

The background art technologies which described in this chapter are well known in the art. The description provided herein is not intended to be exhaustive, but rather to describe those aspects of the background art technologies which are optionally implemented to support the management system of the present invention. Thus, one of ordinary skill in the art could easily use these background technologies in combination with the teachings of the present invention, without requiring undue experimentation.

The preferred background art technologies which are described herein include XML, described in section 1; CORBA and a particular proprietary embodiment of CORBA, described in section 2; and the Java Bean component architecture, described in section 3.

Section 1: XML

One exemplary technology for the implementation of the present invention is XML with ActiveX ™ objects as the front end, such that the information components of the present invention are preferably accessed through XML. The acronym "XML" stands for "Extensible Markup Language". XML is a document markup language which was designed to have greater functionality than HTML (hypertext markup language). Documents written in HTML can, however, be converted into XML.

A document written in XML is a collection of XML elements, which can be images or sections of text, for example. The document itself features elements which are indicated with "tags". These elements have logical values. Each element can also have "child elements", which are other elements to which a reference is made that element, known as the "parent element". This element structure is a hierarchical tree which enables complex elements to be composed of multiple simpler elements.

Documents written in XML optionally feature a DTD (Document Type Declaration), which is either included within the XML document, or alternatively is a separate but associated document. The DTD contains the rules according to which the XML document should be interpreted, such as the declarations for the structures of the elements within the XML document. Hereinafter, the term "HTML" document will refer to a document written in HTML, and the term "XML" document will refer to a document written in XML. The option of including the DTD both increases the flexibility of XML and enables documents written in XML to be validated, in order to ensure that these XML documents conform to the rules in the DTD.

Unfortunately, one drawback of the very flexibility of the DTD structure is that different XML documents may have different associated DTD's, such that similar elements in different files may be described with different "tags". In addition, meta data fields could have different tag names for the same fields. Attempting to locate information within these different documents according to the tag names could therefore be very difficult.

An additional useful feature of XML is the more powerful linking structure available. The links of XML are compatible with those of HTML. However, in addition, XML allows any type of element to act as a link. Furthermore, the start or the finish of an XML link does not need to be located within one of the documents which is being linked. In other words, XML links could be located in a document which is entirely separate from the two linked documents. This enables two documents to be more easily linked after both documents have been created, without altering either document.

There are several different types of XML links, including simple and extended. A simple link is similar to the link of HTML, in that it is unidirectional and has only one locator. An extended link, by contrast, can have more than one locator, such that the extended link can "point" to more than one target resource. Furthermore, extended links can also be bidirectional or multidirectional. As noted previously, these extended links may be located in a separate file, external to the XML document, and can therefore be very difficult to manage. For example, when a linked file is deleted or otherwise removed, the extended link list is not amended, potentially leading to a broken link.

XML links can also point to target resources which are *fragments* of a document. The boundaries of these fragments can be determined through either static "chunking" or dynamic "chunking". For static chunking, the XML file is manually divided into pieces, with a new XML file for each piece which is then linked to the main XML file. For dynamic chunking, the XML file is not divided into new files. Rather, separators are placed within the XML file to indicate boundaries for chunks. These separators can be used to define a portion of a document which is to be retrieved, such that the fragments are separated and served "on the fly". Although it is automatic, dynamic chunking has the disadvantage of significantly increasing server overhead as the server determines which fragment is to be served, such that the XML server may become overloaded.

XML documents also have style sheets, which feature construction rules describing how each element should be displayed. For example, if the element is a paragraph of text, the construction rule may indicate font size and type, the extent of the indentation of the first line, spacing between lines of the paragraph and so forth.

Further information on the XML language is widely available, for example in instructional manuals (Part VIII of *HTML 4 Unleashed - Professional Reference Edition*, Rick Darnell *et al.*, Sams.net, Indianapolis, Indiana, USA, 1998).

Section 2: CORBA

According to an alternative embodiment of the present invention, both the information components of the present invention and the management system for these components are compliant with the CORBA (Common Object Request Broker Architecture) standard, which is a standard for communication between distributed objects established by OMG (Object Management Group). OMG is a consortium of over 700 different software developers. Thus, standards developed by OMG are industry-wide and software applications

compliant with these standards should be able to successfully interact with other compliant applications, as described below.

CORBA is a standard which provides a standard method for execution of program modules in a distributed environment, regardless of the computer programming language in which the modules are written, or the computing platform on which they are executed. CORBA enables complex systems to built, integrating many different types of computing platforms within an entire business, for example.

In order to permit different software applications to communicate, regardless of programming language, hardware or operating system, all such applications communicate through a CORBA-compliant ORB (Object Request Broker). Each application is an "object" with a particular interface through which communication is enabled. ORB acts as the "middle-man", passing information and requests for service to each object as necessary. Thus, one software application does not need to understand or know the interface used by another object, since all communication occurs through ORB.

Furthermore, the use of an ORB permits true distributed computing, since different objects do not need to be operated by the same computer or even reside on the same network. The ORB directs any communication to the appropriate server which contains the object, which might be located on the same host, or a different host, as the client object. The ORB then redirects the results back to the client object. Thus, CORBA can also be described as an "object bus" because it is a communications interface through which objects are located and accessed.

In addition, CORBA provides IIOP (Internet Inter-ORB Protocol), which is the CORBA message protocol for communication on the Internet. IIOP links GIOP (CORBA's General Inter-ORB Protocol) to TCP/IP, the general communication protocol of the Internet. GIOP in turn specifies how one ORB communicates with another ORB. These two types of protocols were implemented to enable different proprietary ORB implementations to communicate over the Internet. Therefore, one type of proprietary ORB can communicate with another, different type of proprietary ORB on a different host computer according to a combination of IIOP and GIOP protocols. Practically speaking, if IIOP is built into a Web browser such as Netscape™ Navigator™, a Java applet is downloaded into the Web browser when the user accesses a Web page with a CORBA-compatible object. The Java applet invokes the ORB to first pass data to the object, then to execute the object and finally retrieve the results.

Further information on both CORBA and IIOP can be obtained from the "TechWeb Technology Encyclopedia" (*http://www.techweb.com/encyclopedia* as of September 10, 1997).

One proprietary version of the CORBA technology for enabling distributed web-based applications is called the Web Request Broker (WRB), developed by Oracle Corp. (Redwood Shores, California, USA). WRB is described in a white paper (M. Anand *et al.*, "The Web Request Broker: a Framework for Distributed Web-based Applications", *http://www.olab.com/www6_1/paper.html* as of September 10, 1). Briefly, the WRB architecture includes the dispatcher, application and system cartridges, and a CORBA compliant ORB. The dispatcher and cartridges use the ORB for communication between components, so that these components can be distributed on separate remote machines. The dispatcher routes requests from the HTTP daemon to the appropriate cartridge. The cartridges are software components which perform a specific function and are thus the "objects" described previously.

Cartridges are used within the system of the present invention as an exemplary support for a number of different functions, as described in subsequent sections. Cartridges have a name, composed of the IP address of the server where the cartridge is located, and the virtual path to the location of the cartridge on that server. Cartridges also have a standard interface, which includes a number of methods. Examples of such methods include the authenticate routine, which determines whether the client is entitled to requested services and the exec routine, which receives the particular service request if the authentication routine is successfully performed. Thus, the cartridge technology provides a fully developed basis for the creation of particular software functionality.

One particular advantage of employing the proprietary cartridge technology for software development is that the system architecture provides a framework for interaction between different objects over the Internet by using HTTP Web servers and existing Web browsers. The CORBA protocols only define a standard, but do not provide any specific implementation. Thus, the proprietary cartridge technology enables one of ordinary skill in the art to develop a software application which can communicate with other applications over the World Wide Web.

Section 3: Java Bean

Another type of enabling background art technology is "Java Bean". Java Bean is a component software architecture which operates in the Java programming environment.

Java, of course, is an interpreter-driven, object-oriented computer programming language which is substantially platform-independent. Software packages which are written in Java can be operated by any operating system, or platform, which supports the Java interpreter. Similarly, a Java Bean component can run remotely and independently as a discrete software application object in a distributed computing environment using either the Remote Method Invocation protocol of Sun Computers Inc., or else by using CORBA. As described below, information components are preferably packaged and then distributed as independent Java Bean components.

The Java Bean component software architecture is a set of API's (Application Programming Interfaces) and rules which enable software developers to define software components to be dynamically combined to create a software application. The Java Bean component model has two major elements: components and containers.

Components range in size and capability from small GUI (graphic user interface) widgets such as a button, to an applet-sized functionality such as a tabular viewer, and even to a full-sized application such as an HTML (HyperText Mark-up Language) viewer or the information component of the present invention. Components can have a visual aspect, such as a button, can actually be visual information or can be non-visual, such as a data-based monitoring component.

Containers hold an assembly of related components. Containers provide the context for components to be arranged and interact with each other. Containers are occasionally referred to as "forms", "pages", "frames" or "shells". Containers can also be components, so that a container can be used as a component inside another container.

The Java Bean component model provides the following major types of services: component interface exposure and discovery; component properties; event handling; persistence; application builder support and component packaging. Component interface exposure and discovery allows components to expose their interface so that they can be driven dynamically by calls and event notifications from other components or application scripts. Component properties are the public attributes of a component which either directly reflect or effect the current state of that component. For example, properties could include the "foreground color" of a video clip, its zoom factor or its access rights. The state of these properties can be interrogated or modified through standard mechanisms.

Event handling is the mechanism for components to "raise" or "broadcast" events and have those events delivered to the appropriate component or components which need to be notified. Typically, notified components then perform a particular function in response. For

example, if the user interface shows a document image clip on the monitor screen, the Parent Information Object event will communicate with the Object Server to transmit the full page of the clip, and will send a viewing command to the full-page viewer component. Thus, event handling allows information components to interact with each other.

Persistence is the mechanism for storing the state of a component in a non-volatile location. The component state is stored in the context of the container and in relation to other components. For example, if the user wants to save the viewing zoom factor for all of the following documents, the persistence mechanism would support this.

Application builder support interfaces enable components to expose their properties and behaviors to application builder development tools. Using these interfaces, the tools can determine the properties and behaviors, or events and methods, of arbitrary components. The tools can provide mechanisms such as tool palettes, inspectors and editors. which the application developer uses to assemble an application. Through these mechanisms, the application developer can modify the state and appearance of components as well as establish relationships between components.

This mechanism enables sophisticated information applications such as HyperText links to be created. For example, using an appropriate multimedia tool, the user can define a button which appears on the viewed document, and then links the document to a different document. The application developer will use property editors to specify the appearance, including size, color and label, of the button, the link type and the link target.

Since Java Bean components can be distributed and independently deployed over a network, there is a need to provide a facility to physically "package" the resources which are included in an information component so that they are accessible to the other Java Bean components. Preferably, such packaging is performed with the JAR (Java Archive) file format. The JAR file format enables the class file of the information component and other information component resources such as images, OMS (object mapping structure), sounds, and link information, to be packaged as a single physical entity for distribution.

## Chapter II: Information Component Architecture

This chapter provides an overall view of the information component architecture and system of the present invention. Section 1 provides a general description of information components. Section 2 describes the system of the present invention. Section 3 describes the information component content capturer in more detail. Section 4 describes the information component identifier. Section 5 describes the information component

contributor. Section 6 describes the information component server. Section 7 describes the information component publisher and client.

Section 1: Information Component

Each information component has a number of different elements and properties. Each information component belongs to an information class. The information class defines the properties and operations of a group of information components. Information classes can describe a newspaper, a general document or a video clip, for example.

Referring now to the drawings, Figures 1A-1C are illustrations of exemplary information components, each of which can be placed in different classes. Figure 1A is a general description of an exemplary document **10**, showing the hierarchical structure. Document **10** is in turn subdivided into a number of page components **12**, of which four are shown for the purposes of illustration only. Page component **12** is a member of the page class, which stores properties related to the structure of a page of document **10**. These properties include textual information, structural information and any links to other components. The operations, or methods, include retrieving the textual information, for example. Thus, the operations are used to store, retrieve or modify information contained in the properties of components which are members of the page class.

Every information component which is a page component **12**, and hence which belongs to the page class, may share certain repeated structural features. These features are also examples of information components, and are described as "shared information components". Every page component **12** can include these shared information components in order to maintain a uniform structure between pages, for example, and to decrease storage space for such repetitive features. As shown in Figure 1A, these shared information components for page component **12** include a footer component **14**, a header component **16** and a logo component **18**. These are intended as examples only and are not meant to be limiting in any way.

Header component **16** could be a title, such as "Document Report", or any other desired information. Footer component **14** could feature a page number, a date, or any other desired information. Logo component **18** would be the logo for the particular company which is producing document **10**, for example. These three information components are shared between all page components **12** which are shown. One advantage of subdividing each page component **12** into various information components is that shared information components, such as footer component **14**, header component **16** and logo component **18** for

example, only need to be stored once on the storage medium or media which holds the information components. Thus, significant savings can be realized in terms of storage space required by sharing repetitive elements between information components.

Each page component **12** also includes one or more information components which are not shared at all, or which are only shared with certain other page components. For example, page component **12** which is labeled "Page 1" includes a summary section component **20**, which is a member of the summary class. The summary class could feature text and/or images which summarize an earlier portion of document **10**, for example. As illustrated in Figure 1A, summary section component **20** is only included within page component **12**.

The "Page 1" page component **12** also features a "Chapter 1" component **22**, which is shared by the "Page 2" and "Page 3" page components **12**. By contrast, the "Page 4" page component **12** features a "Chapter 3" component **24**. Summary section component **20**, "Chapter 1" component **22**, and "Chapter 3" component **24** are all further subdivided into a plurality of paragraph components **26**, which are members of the paragraph class. As their name suggests, paragraph components **26** contain the information related to each paragraph, which may include text for example. The text in each paragraph component **26** is contained within a text component **28** as shown, which belongs to the text class.

In addition, paragraph component **26** can optionally include an image component **30** and a table component **32** as shown. Image component **30**, which belongs to the image class, stores an image, while table component **32** stores a table and belongs to the table class. These three components, text component **28**, image component **30** and table component **32**, are examples of information component primitives. An information component primitive is the most basic unit of information components, such that the primitive is no longer divisible into information components which are lower in the hierarchical structure. Like other information components, information component primitives are preferably potentially able to be shared between information components. For example, a table of data, which is an example of table component **32**, could be included both in summary section component **20** and "Chapter 1" component **22**. As for the shared information components described earlier, shared information component primitives also only need to be stored once in order to be available to other information components.

Figures 1B and 1C show portions of certain specific examples of information components, shown in terms of an exemplary class structure, it being understood that this is for the purposes of description only and is not meant to be limiting in any way.

As shown in Figure 1B, a newspaper information component belongs to a newspaper class **34**, which defines the properties and operations of components which contain newspaper pages. Newspaper class **34** has an article class **36** for an individual newspaper article. Article class **36** inherits the properties of the parent class, newspaper class **34**. In addition, article class **36** may have additional properties and methods, such as the coordinates of the location of the article within the newspaper page, or an operation for retrieving the name of the author of the article. A column **38** is shown for a column, while an image class **40** is also shown for a picture. For example, image class **40** might have information about pictures which are associated with the article. Column class **38** might contain information about the structure of the column which contains the article. Column class **38** and image class **40** are related to article class **36** according to a defined set of relationships.

Figure 1C shows an exemplary video clip information class **42** which contains information such as data and structure for a segment of recorded video. A video stream information class **44** is the highest level class for the hierarchy. A video clip information class **46** is next in the hierarchy, followed by a frame class **48**. Frame class **48** might contain only information regarding a single frame of the video. Thus, even though a video may be considered as a sequential collection of images which give the illusion of movement, it too can be broken down into smaller elements which are then stored in the above-mentioned information classes.

Section 2: General System Architecture

This section provides an overview of the general architecture of the management system of the present invention, as well as of the interactions between the four main elements of this system. The specific functions of each element will also be described in successive sections below.

Figures 2A and 2B show the general architecture of the system of the present invention. As shown in Figure 2A, a general system architecture **50** includes IC Contributor **60**, IC Server **62**, IC Search Engine **63**, and IC Publisher **65**. As shown in Figure 2B, IC Contributor **60** further features IC (Information Component) Content Capturer **52**, IC Knowledge Base **54**, IC Rules Editor **56**, and IC Identifier **58**.

Although each element will be described in further detail below, briefly IC Content Capturer **52** is responsible for the acquisition and conversion of information content, and for the transmission of the converted information content to IC Identifier **54**. IC Identifier **54** then identifies information components according to certain rules and to class information

stored in IC Knowledge Base **54**. Both the rules and the class information can be added, removed or otherwise altered with IC Rules Editor **56**.

Once the information components have been identified, the original document and the identified information components are transmitted from IC Contributor **60** to IC Server **62**. IC Server **62** then stores and manages the actual or "original" information such as documents, multimedia objects and other types of information entities, as well as managing the information components themselves. Information components are made available from IC Server **62** by a request through IC Search Engine **63**, and are then published by IC Publisher **65**. Thus, the general system of the present invention collects the information from a variety of sources, packages the information into information components, and then stores the components for later retrieval by a client application.

Section 3: Information Component Content Capturer

This section describes the IC (information component) Content Capturer, which is shown in Figure 2B, as part of IC Contributor **60**.

IC Content Capturer **52** preferably operates as memory resident software and captures the desired information content from a variety of software systems including, but not limited to, a document editor **64** such as the Word product of Microsoft™; a media application **66** including, but not limited to, the Adobe™ Acrobat™ reader for reading PDF files from Adobe™ Acrobat™; a facsimile machine software application **68** for operating a facsimile machine; and a Web browser software application **70** such as Netscape™ Navigator™. Additional software systems from which information content can be captured include imaging software and spreadsheet software. These software systems are intended as illustrative examples only, since substantially any software system which handles, stores, retrieves or manipulates information could have that information captured by IC Content Capturer **52**.

IC Content Capturer **52** invokes the appropriate software drivers for handling different information formats from the above software systems. As an example, information could be captured from a document stored in the format of Microsoft™ Word™ word processing software. A number of possible methods could be used to capture the information contained within the document, two illustrative examples of which are given here, it being understood that these are for discussion purposes only and are not meant to be limiting.

In the first exemplary implementation, IC Content Capturer **52** interacts with Microsoft™ Word™ and instructs Microsoft™ Word™ to place the document on the "clipboard". The "clipboard" is a feature of a number of different computer operating systems, in particular those operating systems of Microsoft Inc. (Seattle, Washington, USA), such as "Windows95™" and "Windows NT™", for example. The general function of the "clipboard" is to enable one software application, such as Microsoft™ Word™, to make information available to another software application, such as IC Content Capturer **52**. Hereinafter, the term "clipboard" refers to any feature of a computer operating system which enables information to be exchanged between two software applications. Once the document has been copied to, or placed on, the clipboard, the document is then pasted to IC Content Capturer **52**.

In the second exemplary implementation, IC Content Capturer **52** captures the necessary information about the document through substantially direct interaction with the software system, such as Microsoft™ Word™. Such interaction can be performed according to a number of different methods. For example, Microsoft™ Word™ enables other software applications to obtain this information through the creation of a "macro". Alternatively, IC Content Capturer **52** could include a printer driver, which would enable Microsoft™ Word™ to "print" the document to IC Content Capturer **52** directly, or alternatively to a file in a format accessible by IC Content Capturer **52**.

In any case, regardless of the specific method employed, the content of the information is obtained from the captured information by using a particular software driver. Each software driver is relevant to the particular information source format, such as electronically scanned paper document, electronic document such as a word processing document, video clip, document sent by facsimile and other such formats. Each driver is a channel to an information processing unit for a specific type of information, and invokes a process specific to the source of that information. Finally, the content information is stored in an internal unified format for data processing and information component recognition, access and retrieval. The information in the unified internal file format is then sent to IC Identifier **58**.

Section 4: Information Component Identifier and Knowledge Base

IC Identifier **58** automatically identifies and creates information components from the information passed from IC Content Capturer **52** according to rules stored in IC Knowledge Base **54**. As an example, preferably the information is first analyzed to extract the

information component primitives, which as described previously form the most basic unit of information. These primitives include text, images, vector graphics and other such basic units of information. Next, the information components themselves are constructed from the information component primitives, and the relationships between components are determined according to rules stored in IC Knowledge Base **54**. Finally, the information components are classified, again according to rules stored in IC Knowledge Base **54**. This classification determines security attributes, indexing rules and other publishing parameters. At the end of this stage, the information is transferred to IC Server **62**.

Figure 3A shows a portion of IC Contributor **60** in more detail, focusing on those components which interact with IC Identifier **58**. IC Identifier **58** has three layers, including a primitive identifier **64**, a component constructor **66** and a component classifier **68**.

Primitive identifier **64** examines the received information at two levels. First, the textual information is identified and separated into individual elements, according to the structure of the type of information. The second level of examination of the received information is visual identification, which includes determining the visual attributes and structure of the information. At the end of this dual level examination, the information component primitives have been identified according to rules stored in IC Knowledge Base **54**.

The information component is then constructed from one or more information component primitives and/or from one or more information components which are lower in the hierarchy, by component constructor **66**. The information component includes such information as the identity of the primitive(s) or lower information component(s) from which it is constructed. In addition, the relationships between components are determined by component constructor **66** according to rules stored in IC Knowledge Base **54**.

An illustrative example of this process is disclosed in U.S. Application No. 08/318,044, herein incorporated by reference. The disclosed process includes the following steps. First, the document is converted into a digital raster format, for example by scanning a paper document, which is stored in an electronic file. This step is preferably performed by IC Content Capturer **52**. Next, preferably the converted document is enhanced to improve the quality of the image, for example.

In the third step, the enhanced raster format file is converted into two electronic files, collectively called a "binary/raster file". The first file has the enhanced raster format, while the second file has pointers to the enhanced raster format file. Every data element in the raster format file, such as textual information or an entire graphic image, could have a

corresponding pointer in the second file. Thus, the two files are preferably produced, at least in part, by an automatic text recognition process such as OCR, which enables the image of the text to be realized as textual data. The information is then stored as information components composed of information primitives, as previously described.

Once all of the data has been placed in a database as information components, indices for information retrieval are created. Thus, the original document has been subdivided and stored as a collection of information components.

These information elements preferably include a raster image of the document, a pointer to the storage location of the original document, any text contained within the document and the coordinates of the words of the text within the document. More specifically, the coordinates preferably include all information which is necessary to geographically locate the word within the document, such as the number of the page on which the word falls, the number of the word on the page and the coordinates of the rectangle which bounds the word on the page, or "bounding rectangle". The bounding rectangle determines the area occupied by the word on a page and is necessary to fully reproduce the visual aspects of that word. Thus, the coordinates of each word numerically describe the visual appearance of the word.

As an example of this process of obtaining text and coordinates, if the source of information is a paper document which has been scanned to an electronic file, IC Content Capturer **52** performs OCR (Optical Character Recognition) to obtain the textual information from the image stored in the electronic file by converting the image of a letter into the letter itself. Both the text itself and the coordinates of individual words are then available. Other examples of such processes include pattern recognition and PDF conversion. It should be noted that these processes are already well known in the art for the creation and manipulation of information in a particular information source format.

The information elements which are produced are then identified according to the type of information component primitive which they represent, which is in turn determined according to rules in IC Knowledge Base **54**. For example, every individual image identified in the steps above would be determined to be an image information component primitive. Similarly, the text extracted in the steps above would be determined to be a text information component primitive according to information stored in the textual database. Other information component primitives could also be identified from the collection of information elements.

After the information component primitives have been identified, the primitives are used to construct information components, according to rules stored in IC Knowledge Base **54**. For example, in document component **10** of Figure 1A, the information primitives include image information component primitive **30** and text information component primitive **28**. These primitives are in turn used to build paragraph **26**. Paragraph **26** now contains information concerning not only the inclusion of one or more image information component primitives **30**, for example, but also such information as the relative geometrical location of the primitive within paragraph **26**. The geometrical location of the primitive was determined when the primitive itself was identified, for example as described above. Thus, the primitives are first assembled in information components which are relatively lower in the hierarchy, for example paragraph **26**, and then these components are in turn assembled into information components which are higher in the hierarchy.

Either after the individual components have been constructed, or substantially simultaneously during the construction process, each individual information component is classified according to rules in IC Knowledge Base **54** by component classifier **68**. The individual component is compared to components listed within the knowledge base, and is recognized as a unique and individual element belonging to a larger information cluster. Each component is classified first by assignment to a primary information class, and then by placement within the hierarchical structure of information sub-classes belonging to that primary class.

As shown with reference to Figure 3B, which shows a schematic block diagram of an exemplary IC Knowledge Base **54**, first the document class for the information component is determined. The different document classes are stored within IC Knowledge Base **54** in a document class table **70**. For example, the document could be a research report, newspaper, or substantially any other type of document which has been placed within document class table **70**.

Next, the information component would be classified according to a particular information component class stored in an IC class table **72**. These classes could include, but are not limited to, a logo, a main title, publishing information, summary, and so forth. Each class is in turn identified according to rules stored in a rules table **74**. These rules are composed of tokens, including constants **76**, functions **78** and operations **80**. Each rule could optionally be stored in a "flat (text) file" for example, in which case the tokens would preferably be stored as text strings separated by spaces. Of course, many other options are also available for storing these rules.

Each rule preferably includes the following tokens in the following order, although of course other rule structures could be used: the name of the IC class, the hierarchy level of the information component, the font type, the size range, the color, the case of the letter, the location of the page on which the information component is found and the text which the information component should contain (if any). The rule does not necessarily need to include all of these tokens; an absent token can be indicated by a place-holding character such as a "slash" ("/"), for example.

One example of such a rule is "PageNo -1 Helvetica-Bold 11.0-11.05 / / B /". According to this rule, an information component, named *PageNo*, is identified by any text of any color in any letter case, in font *Helvetica-Bold*, any size between 11 and 11.05, which is located at the bottom of the page (indicated by the letter "B").

As another example, consider the rule "Section 1 TimesNewRoman/TimesNewRoman-Bold 9.50-10.50 / / / /". According to this rule, an information component, named *Section*, is identified by any text of any color in any letter case, in font *TimesNewRoman* or *TimesNewRoman-Bold*, any size between 9.50 and 10.50, which is located anywhere on the page.

The rules and other information stored in IC Knowledge Base **54** are optionally and preferably edited through an IC Rules Editor **56**. IC Rules Editor **56** is preferably a GUI (graphical user interface), which more preferably allows the user to define new rules, enter new information, delete old rules or information, and amend or alter rules or information.

Once the information components have been constructed and classified, they are passed to the IC Server for being served.

Section 5: Information Component Contributor

As shown in Figure 4, IC Contributor **60** also prepares the information components for publication and for storage in a database, such that the information components can be served to a client by IC Server **62**. IC Contributor **60** features a component generator **82**.

Component generator **82** transforms the classified information component into a standard format including, but not limited to, an active object format such as a COM object or a Java Bean object, or a flat file format such as DHTML.

Generally, component generator **82** packages the classified information component according to the standard format, so that the packaged information component is accessible by IC Server **62**. For the purposes of discussion only and without any desire to be limiting, descriptions of the transformation of the information component into two of these object-oriented formats are given in further detail below. In Chapter III, Section 2, a description is

provided of the transformation of the information component into an object in an XML environment. In Chapter IV, Section 1, a description is provided of the transformation of the information component into a Java Bean object in a CORBA environment.

IC Contributor **60** is also able to render and to store information components as a DHTML document. In this embodiment, IC Contributor **60** converts data for each primitive of each information component into equivalent fragments in DHTML format. There can be two types of data elements in the source information component: graphic elements and text elements. The graphic elements are converted to raster images (in GIF format).The text elements are converted to a set of DHTML <DIV> blocks.

There are two types of DHTML <DIV> blocks: a style block and a value block. The "style block" defines the style attributes of the text, such as font size and name, font-weight, font-style and color. The "value block" defines the position of the text element within the current primitive and its text value. When the text value contains more than one word, the text value is inserted into a <NOBR> block to prevent line breaking for the given text element by the web browser.

To minimize the size of the result DHTML fragment for each primitive, the DHTML fragment is optimized to ensure that each "style block" with specific characteristics appears only once in DHTML fragment for the primitive.

Exact correspondence between the source document text style and the DHTML style is not always possible. In this case, the original fonts are preferably substituted by the fonts available for the Web browser with possible modifications of font size.

IC Server **62** can then serve the information component to IC Search Engine **63** after receiving a request for a particular information component from IC Search Engine **63**. As described in further detail below, IC Search Engine **63** receives a request for an information component, which is then made available to IC Publisher **65**, which publishes the information component. Optionally, IC Publisher **65** publishes the information component to a Web page, for example, or onto paper, as another example.

Both IC Search Engine **63** and IC Server **62** must be able to communicate with each other, such that an information component can be requested. This communication is permitted with information components which have a standard format. An object format is particularly preferred, because objects can be accessed through a predefined structure, which is more efficient for interacting with the information contents of the object. Both of the exemplary and preferred embodiments described below in Chapter III, Sections 1 and 2

(XML) and in Chapter IV, Section 1 (Java Bean) have object formats for the information component. Of course, other types of formats could be used, such as DHTML.

Section 6: Information Component Server

Once the information component has been transformed into an active object or other general format by component generator **82**, the active object and the original document are then sent to IC (Information Component) IC server **62** and are then stored in a database **84**, as shown in Figure 5. IC server **62** stores and manages the "original" information, such as documents, video segments, sounds and so forth. When a client application issues a request for information, IC Server **62** locates the original information entity, isolates the corresponding information component and then returns the information component to the client application in some suitable format, for example as an HTML file.

Section 7: Information Component Publisher and Client

As shown in Figure 6, an IC Client **98** is able to send requests for information components to IC Server **62** through an IC Search Engine **63**. In addition, IC Client **98** is also able to receive such components from IC Server **62**, optionally and preferably through the CORBA ORB. IC Client **98** preferably features some type of GUI (graphical user interface), which enables client applications to interact with the functionality of the information management system of the present invention. IC Publisher **65** is then able to publish the information component onto IC Client **98**.

In preferred embodiments of the present invention, the ability to access certain information components and to view these components on GUI interface **100** is controlled by two functions: automatic information component replacement and "white-label". These features provide customized views of the same documents to different user groups, while preventing the display of sensitive information components to specific users or to groups of users.

Automatic information component replacement is accomplished through an IC replacement table, which is preferably stored in database **84**. The IC replacement table includes the following information: the class and the name of the information component to be replaced, the class and the name of the information component which is to replace it, and the user's groups or individuals for whom the replacement should be performed. For example, the logo on a particular research report, which is an information component called "Big Company X Report", could be replaced by the logo of "Another Big Company" for the

user's group "Another Big Company". IC Server **62** would then replace the logo of "Big Company X" with the logo of "Another Big Company" when those clients of "Another Big Company" request the Report.

The white-label function is used to specify one or more information components in the original document which are not to be displayed on GUI interface **100**, but which remain incorporated within the original document. Thus, the white-label function enables sensitive information to be protected from access through IC client **98**.

## Chapter III: Specific Exemplary Implementation with Objects in an XML Environment

This chapter describes the details of an exemplary preferred embodiment which implement the system of the present invention with objects in an XML environment. The objects are preferably compatible with the ActiveX™ architecture, although other types of objects could also be used, as long as they were compatible with the XML environment. For example, the ActiveX™ objects could be constructed by the client from the information component objects according to the ActiveX™ architecture.

The list of sections in this chapter is as follows. Section 1 is an overview of the system when implemented with XML. Section 2 is a description of IC Contributor when implemented with XML. Section 3 describes IC XML server. Section 4 describes the IC Search Engine when implemented with XML. Section 5 describes the IC Publisher and IC Client when implemented with XML.

### Section 1: Overview of System with XML

Figure 7 shows an overall view of a portion of the system of the present invention as implemented in XML. IC Contributor **60** is now IC XML Contributor **200** and IC Server **62** is now IC XML Server **202**. IC XML Contributor **200** creates objects from the information components as XML-environment compatible objects.

IC XML Server **202** provides access to a database **204**, which is similar to database **84** of Chapter II. Database **204** stores the information components, which are implemented as XML-environment compatible objects.

IC XML Server **202** also communicates with a DOM (document object model) compliant interface, referred to as DOM Interface **208**. Software programs which are compliant with the DOM protocol are able to communicate with other software programs for XML-compatible or XML-specific tools, such as Web browsers or software programs for

editing XML documents, for example. Thus, DOM Interface **208** acts as a gateway, enabling these XML tool software programs to communicate with information components through IC XML Server **202**. As described in greater detail in Section 3 below, XML tool software programs can therefore preferably edit and reuse information components directly from database **204**, without conversion of the components.

IC XML Server **202** provides one or more information components upon receiving a request from IC Universal Search Engine Adapter **214**. IC Universal Search Engine Adapter **214** enables many different types of search engines to communicate with IC XML Server **202**, such that a search can be made for specific information components within database **204**. IC Universal Search Engine Adapter **214** also preferably controls access to IC XML Server **202**, preferably including such functions as security and request access. IC Universal Search Engine Adapter **214** passes a request for an information component to IC XML Server **202**, which then returns the desired information component. One or more information components can then be served to IC XML Publisher **210**.

**IC XML Publisher 210 optionally and preferably includes an HTML rendering engine 212, and a standard document rendering engine 216. The information component can** then be displayed to the user in a number of ways, such as by printing the information on paper or by displaying the information on a Web page.

If the information is to be printed on paper, or otherwise rendered in a "standard" document format, then IC XML Publisher **210** passes the information component to standard document rendering engine **216**. If the information is to be displayed by a Web browser which can only handle HTML documents, then IC XML Publisher **210** passes the information component to HTML rendering engine **212**. Other types of rendering are also possible of course.

The description of each of these parts of the system is given in greater detail in the sections below.

Section 2: Specific Exemplary Implementation of IC Contributor with XML-Environment Compatible Objects

Section 5, Chapter II above described the general implementation of IC Contributor **62**. This section describes a specific, preferred implementation of the IC Contributor for operation with XML-environment compatible objects such as those compatible with ActiveX™ architecture, IC XML Contributor **200**.

As XML-environment compatible objects, information components are organized in a hierarchical structure and linked to each other. Each XML-environment compatible object has methods, properties and data. The data itself is the classified information component obtained as described in Chapter II.

Methods determine the ways in which the data and properties of the information component can be manipulated. For example, methods include ways to access the data, whether as an image, a video clip, a sound and so forth. Methods also include an application interface, so that another application would be able to interact with the information component and with the stored data, and with a GUI (graphical user interface). Other methods pertain to access control and to event handling. Event handling enables these objects to broadcast events and to have those events delivered to an appropriate component or components for notification. Thus, event handling provides methods for communication between components packaged as XML-environment compatible objects.

Although individual methods might be specific to a particular information component, such that a newspaper article component would probably not include a method for manipulating sound, the overall mechanism for describing each method is supported by the object architecture and could be easily determined by one of ordinary skill in the art.

The properties of the XML-environment compatible object include the internal structure of the object and the location of the data of the information component within the hierarchical structure of information components. For example, as described in Section 4, Chapter II, information components are composed of IC primitives, which are in turn used to build more complex structures which describe the relationships between information components. The location of the data of the information component within a hierarchy is important in order to be able to construct virtual documents and to understand the type and significance of the data within the information component.

In addition, these properties include the correct tags for the type of data within the object, in order for the object to be correctly rendered within the XML environment, and its location within the information component hierarchy. For example, if the type of data is a chapter of a book, then the correct tag might be the "chapter" tag. This tag identifies the type of XML element for the object, which is important for the later assembly of the data within the object as an element of an XML document.

IC XML Contributor **200** packages the information component obtained from IC Identifier **58** into the XML-environment compatible object as follows. First, the data of the information component forms the data of the object. Next, the methods which can be used to

interact with the object are determined. Certain of these methods are typical for all such objects. Other methods, such as the method for accessing the type of data within the object, are particular for the type of data from the information component. Finally, the properties of the object are determined, for example according to the location of the data of the XML-environment compatible object within the information component hierarchy.

Section 3: Specific Implementation of IC Server

This section describes a specific implementation of IC Server **62**, described in Chapter II, Section 6, for operation with XML-environment compatible objects. IC XML Server **202** accepts requests for and then serves information components as XML elements assembled into an XML document. In addition, IC XML Server **202** manages the extended links of XML and normalizes the structure of various DTD's for the XML documents. Also, in conjunction with DOM Interface **208**, IC XML Server **202** enables the XML-environment compatible objects to be accessed by XML tool software programs without requiring conversion of the objects.

As noted in Chapter I, Section 1, XML documents are collections of one or more XML elements which are organized according to certain rules, which are held in the DTD of the XML document. IC XML Server **202** is preferably able to assemble XML documents "on the fly" in response to a request from a client application.

For example, a client application might request a particular chapter of a book. This chapter could contain a chapter title, text and images, for example. The chapter could also be further subdivided into sections, each of which would also have an organizational structure. The data required to assemble the chapter is contained within one or more IC XML elements. If there are a plurality of IC XML elements, then these elements are related as part of a hierarchy based upon the information component hierarchy of the chapter. Therefore, IC XML Server **202** must first locate all of the IC XML element(s) which are required for the chapter.

Next, a style sheet is optionally selected for the XML document. The style sheet is optionally determined by the properties of the "chapter" IC XML element, which may indicate a particular style sheet to be used for that element. Alternatively and preferably, the style sheet could be determined according to specifications submitted by the client application, such that the preferences of the client application determine the style sheet.

The IC XML elements are then assembled in the XML document, optionally according to the style sheet. The DTD for the XML document is then constructed, according to the tags contained within the IC XML element.

The links for the XML document are then determined. Preferably, these links are extended links. More preferably, the extended links are managed as part of a document which is external to the XML document. These links are determined according to the link(s) of the IC XML element, which are included in the properties of the element. For example, one such link might link two sections of the chapter. Preferably, if a link is to another XML element which is not part of the XML document being assembled, such as for a different chapter, then this other XML element is also assembled into a different XML document, such that the different XML document could also be served if necessary.

A preferred embodiment for link management is described with regard to Figure 8. Preferably, extended links are also objects which are stored externally, for example in database **204**. Extended link objects are exposed as child objects of the IC XML-environment compatible objects, or resource objects, which they link.

The identity of each extended link object is preferably stored in a link table **218**, which is then stored in database **204**. The identity of each IC XML-environment compatible object is also stored in an IC table **220**, also stored in database **204**. Optionally and more preferably, a document table **222** is also stored in database **204**. Document table **222** indicates how to assemble complete XML documents. Preferably, these XML documents can be assembled into a format which closely resembles the format of the original document from which the information was obtained. Also preferably, other "virtual" XML documents could also be assembled according to requests received from the client application.

IC XML Server **202** manages the extended link objects through dynamic management, by dynamically generating extended link objects as required. For example, if an document or an information component is removed from database **204**, IC XML Server **202** updates link table **218**, IC table **220** and document table **222**. Other changes to the link structure may occur manually, through a link editor **224**. Alternatively and preferably, changes to the link structure may occur when a document is edited, added or deleted through a document manager **226**. An XML editor **228** may also be used to change the structure of the links. IC XML Server **202** updates link table **218**, IC table **220** and document table **222** as necessary. More preferably, IC XML Server **202** sends an alert to the software tool which is attempting to remove the document or information component from database **204**, alerting the user to the possible alteration to the link structure.

Once the XML document has been prepared, it is sent to IC XML Publisher **210**, for example for being published as a Web page. IC XML Publisher **210** is described in greater detail in Section 4 below.

With regard to DTD normalization, IC XML Server **202** is preferably capable of serving many different types of XML documents, which may have different DTD structures. Such different structures can increase the difficulty of searching, retrieving and assembling IC XML elements. Furthermore, if IC XML elements have different names for tags which should indicate the same element, IC XML Server **202** may not be able to assemble IC XML elements correctly. Therefore, IC XML Server **202** optionally and preferably performs DTD normalization for the XML elements and documents.

The process of DTD normalization is shown with regard to Figure 9. First, a DTD **230** is received by IC XML Server **202**. IC XML Server **202** then passes each tag within DTD **230** to a DTD normalizer **232**. DTD normalizer **232** compares the name of the tag (text string associated with the tag) to the rule or rules of a DTD rules database **234**. For example, if the name of tag is "summary", a rule might state that "synopsis" should be used to replace "summary". Preferably, if DTD rules database **234** does not have a rule for the name of that particular tag, then DTD normalizer **232** searches any information associated with the XML element having that tag in order to normalize the name of the tag.

With regard to DOM Interface **208**, XML tool software programs, such as editor programs for XML documents, are able to communicate with IC XML Server **202** through this "gateway" software module. For example, these editor programs are able to create and manipulate virtual documents from XML-environment compatible objects stored in database **204** in a substantially similar manner to the way in which XML documents are created and manipulated.

Section 4: Implementation Of IC XML Search Engine

As previously described, IC Universal Search Engine Adapter **214** passes requests for information components to IC XML Server **202**. IC Universal Search Engine Adapter **214** therefore controls access to IC XML Server **202**, and hence to the information components.

IC Universal Search Engine Adapter **214** preferably operates according to an HTTP-based protocol. Preferably, the access offered through IC Universal Search Engine Adapter **214** can be determined according to a software module or applet written in Javascript, Java, Active-X™ or C++ for example. IC Universal Search Engine Adapter **214** is preferably able to translate substantially any type of search query language into a format which is accessible

to IC XML Server **202**. More preferably, IC Universal Search Engine Adapter **214** includes a driver (not shown) for each search engine, such that a new type of search engine can be easily accommodated by altering the driver.

IC Universal Search Engine Adapter **214** is preferably built to be compatible with the particular architecture of IC XML Server **202**, such that the client application requesting a particular information component would not need to be altered in order to be compatible with different search engines.

Section 5: Specific Implementation of IC Publisher

For this embodiment of the system of the present invention, IC Publisher **63** is IC XML Publisher **210**. IC XML Publisher **210** makes the information components accessible to the client application. The information component can then be displayed to the user in a number of ways, such as by printing the information on paper or by displaying the information on a Web page.

If the information is to be printed on paper, or otherwise rendered in a "standard" document format, then IC XML Publisher **210** passes the information component to standard document rendering engine **216**. Standard document rendering engine **216** could output the information component according to the PostScript protocol for example, in order to allow data exchange and communication with paper printing devices.

If the information is to be displayed by a Web browser which can only handle HTML or DHTML documents, then IC XML Publisher **210** preferably passes the information component to HTML rendering engine **212**. Other types of rendering are also possible of course.

HTML rendering engine **212** is able to render the XML document as an HTML or a DHTML document for being served to a Web browser. In addition, preferably HTML rendering engine **212** is able to render other document formats, such as PDF, word processing and image formats, as HTML documents as well. PDF could be rendered from a PostScript output for example. Preferably, the functions described for HTML rendering engine **212** could be used for rendering substantially any type of file in substantially any format as an HTML or DHTML document, as described below.

Figure 10 is a schematic, block diagram showing a preferred implementation of HTML rendering engine **212** and associated items according to the present invention. HTML rendering engine **212** interacts between a native file format processor **236** and a Web browser **238**. Essentially, HTML rendering engine **212** enables a native file format document **240**, which would normally be substantially accessible only to native file format processor **236**, to

be displayed by Web browser **238**. Furthermore, the display of native file format document **240** by Web browser **238** is visually similar or identical to the display of native file format document **240** by native file format processor **236**, as enabled by HTML rendering engine **212**.

Native file format processor **236** can be any software component or application which can access a native file format document **240**. Examples of such software components or applications include, but are not limited to, an XML editing software program, word-processing software such as Microsoft™ Word™ and exchange format software such as Adobe Acrobat™ Exchange/Reader. Examples of native file formats include, but are not limited to, the XML format, the DOC format for Microsoft™ Word™ and the PDF format for Adobe Acrobat™. The phrase "access a native file format document" is meant to connote that native file format processor **236** can display and manipulate native file format document **240** such that native file format document **240** is viewable with native visual attributes or visual appearance. Although a number of "exchange" formats are available, such as the Rich Text Format (RTF), which can be accessed by more than one type of word processing software for example, information is often lost through conversion to and from these formats. Furthermore, these formats are not the native file format itself, but only an approximation. Thus, native file format document **240** is preferably in the file format which is intended to be implemented by native file format processor **236**.

HTML rendering engine **212** is able to convert native file format document **240** into a raster image in a raster format which is displayable by Web browser **238**, according to one of two preferred embodiments of the present invention. In the first embodiment, HTML rendering engine **212** interacts with native file format processor **236** to obtain data regarding native file format document **240**. In the second preferred embodiment, HTML rendering engine **212** directly accesses native file format document **240** substantially without any interaction with native file format processor **236**.

The first preferred embodiment of the present invention has many different possible implementations, two illustrative examples of which are given here, it being understood that these are for discussion purposes only and are not meant to be limiting.

In the first exemplary implementation, HTML rendering engine **212** interacts with native file format processor **236** and instructs native file format processor **236** to place native file format document **240** on the "clipboard" (not shown). The "clipboard" is a feature of a number of different computer operating systems, in particular those operating systems of Microsoft Inc. (Seattle, Washington, USA), such as "Windows95™" and "Windows NT™",

for example. The general function of the "clipboard" is to enable one software application, such as native file format processor **236**, to make information available to another software application, such as HTML rendering engine **212**. Hereinafter, the term "clipboard" refers to any feature of a computer operating system which enables information to be exchanged between two software applications. Once native file format document **240** has been copied to, or placed on, the clipboard, native file format document **240** is then pasted to HTML rendering engine **212** as a graphical image. Thus, HTML rendering engine **212** imports, or accesses, native file format document **240** as an image, which can then be converted to a raster image in a raster format. Additionally, HTML rendering engine **212** is able to obtain the necessary data about native file format document **240** through such "pasting".

In the second exemplary implementation, HTML rendering engine **212** receives the necessary information about native file format document **240** through substantially direct interaction with native file format processor **236**. Such interaction can be performed according to a number of different methods. For example, Adobe Acrobat™ allows other software applications to obtain this information through the creation of a "plug-in". Microsoft™ Word™ enables other software applications to obtain this information through the creation of a "macro". Alternatively, HTML rendering engine **212** could include a printer driver, which would enable native file format processor **236** to "print" native file format document **240** to an image format file. Such "printing" would also give HTML rendering engine **212** the necessary data about native file format document **240**. Thus, HTML rendering engine **212** would obtain the necessary data about native file format document **240** through interaction with native file format processor **236**.

The second preferred embodiment of the present invention has many different possible implementations, one illustrative example of which is given here, it being understood that this example is for discussion purposes only and is not meant to be limiting. As noted previously, the second preferred embodiment involves direct interaction of HTML rendering engine **212** with native file format document **240**, substantially without any interaction with native file format processor **236**.

Such direct interaction has a number of advantages, in particular greater speed of conversion of native file format document **240** to the raster format. HTML rendering engine **212** preferably performs such interaction by understanding all or substantially all of the instructions contained within native file format document **240**, in a similar or identical manner as native file format processor **236**. These instructions are like any another computer

software language, and as such can be understood and interpreted by software applications other than native file format processor **236**.

Regardless of the particular implementation by HTML rendering engine **212**, HTML rendering engine **212** obtains the necessary data about native file format document **240**. This data includes substantially all of the words of the text in native file format document **240**, or at least of that portion of native file format document **240** which is to be displayed on Web browser **238**. In addition, the data includes the coordinates of each word within native file format document **240**. Finally, the data preferably includes all attributes of each word and of the relationships between words, such as the font style and size, character attributes such as bold or italicized text, and spaces between characters and words. Thus, the data in combination enable native file format document **240** to be reproduced in a substantially identical or identical document appearance on Web browser **238**.

More specifically, the coordinates preferably include all information which is necessary to geographically locate the word within native file format document **240**, such as the number of the page on which the word falls, the number of the word on the page and the coordinates of the rectangle which bounds the word on the page, or "bounding rectangle". The bounding rectangle determines the area occupied by the word on a page and is necessary to fully reproduce the visual aspects of that word. Thus, the coordinates of each word numerically describe the visual appearance of the word and, preferably in combination with the visual attributes of the word, enable the visual appearance of the word to be reproduced.

Once HTML rendering engine **212** has received the data from native file format document **240**, HTML rendering engine **212** creates the raster image in a raster format which is displayable by Web browser **238**. The raster image is created from the data obtained from native file format processor **236**, and preserves substantially all of the visual attributes of native file format document **240**, or a portion thereof, when seen in the native document appearance. The raster format is supported by Web browsers. One example of such a format is the GIF raster format. Thus, the raster image, containing at least a portion of native file format document **240**, is displayable by Web browsers.

The raster image is optionally created "on the fly". Alternatively, such a raster image could be stored in an additional database **242** containing cached raster images, rather than being created "on the fly".

If the raster image is produced as a result of a search request by the user, then preferably at least one "match" or search result is displayed in the context of at least a

portion of at least one native file format document **240** containing the match, as shown in Figure 11.

Figure 11 shows an exemplary, illustrative depiction of a portion of the computer monitor screen which is displaying the raster images of two matches. A monitor screen **244** is displaying a portion of the graphic output of Web browser **238**, here shown as Netscape Navigator™ although substantially any Web browser could be used. At the left-hand portion of monitor screen **244**, a command area **246** enables the user to enter commands to HTML rendering engine **212** through Web browser **238**. At the right-hand portion of monitor screen **244**, a display area **248** shows a portion of the results from the search. Display area **248** shows a portion of two documents **250** with the searched term, "Keppel", emphasized, in this example by a box. As can be seen, both graphic images and text are displayable in display area **248**. Thus, the results of the search are displayed by Web browser **238**, preferably within the context of at least a portion of the original document, as shown.

In a preferred embodiment of the present invention, the list of matches includes a plurality of matches within a single native file format document **240**, a single match from a plurality of native file format documents **240**, or even a plurality of matches from a plurality of native file format documents **240**. Web browser **238** can then request the next match in the series of matches, or else the previous match in the series. Again, HTML rendering engine **212** then creates the raster image of the desired match in the series. Again, alternatively such a raster image could be stored in database **242**, rather than being created "on the fly". In any case, the raster image of the desired match is transferred to, and then displayed by, Web browser **238**.

In a second embodiment of HTML rendering engine **212**, HTML rendering engine **212** is able to render information components as a DHTML document. In this embodiment, HTML rendering engine **212** converts data for each primitive of each information component into equivalent fragments in DHTML format. There can be two types of data elements in the source information component: graphic elements and text elements. The graphic elements are converted to raster images (in GIF format).The text elements are converted to a set of DHTML <DIV> blocks.

There are two types of DHTML <DIV> blocks: a style block and a value block. The "style block" defines the style attributes of the text, such as font size and name, font-weight, font-style and color. The "value block" defines the position of the text element within the current primitive and its text value. When the text value contains more than one word, the text value is inserted into a <NOBR> block to prevent line breaking for the given text

element by the web browser.

To minimize the size of the result DHTML fragment for each primitive, the DHTML fragment is optimized to ensure that each "style block" with specific characteristics appears only once in DHTML fragment for the primitive.

Exact correspondence between the source document text style and the DHTML style is not always possible. In this case, the original fonts are preferably substituted by the fonts available for the Web browser with possible modifications of font size.

DHTML representations can be created for complete pages as well as for parts of any page. In order to generate a DHTML view of a document page, the relevant primitives are obtained. These include the DHTML data, the enclosing rectangle for the primitive, and text coordinate mapping (for drawing search "hits" or results, or for otherwise highlighting or emphasizing a portion of text).

HTML rendering engine **212** iterates over these primitives and for each one generates the DHTML code that locates it in the proper place on the page. Graphical elements are handled by creating a combination of a <DIV> and <IMG> tags which point to a URL for loading the images directly.

The search hits are also displayable as part of a DHTML view. Hits are created by adding (prior to the primitive DHTML) <DIV> and <IMG> tags which point to the URL of a pre-defined small image containing the hit color. The hits can be indicated by using one of 3 coloring methods. These include marking the word; marking the beginning of the line; and marking the entire line. The size of the coloring which indicates the hit can be adjusted to the proper size by using the text coordinate mapping of the primitive.

## Chapter IV: Specific Implementation with Java Bean and CORBA

The preceding chapter described the details of implementing the system of the present invention with objects in an XML environment. In this chapter, a description is provided for implementation with Java Bean objects in a CORBA environment.

### Section 1: Specific Implementation of IC Contributor with Java Bean Objects

Section 5, Chapter II above described the general implementation of IC Contributor. This section describes a specific, preferred implementation of IC Contributor for operation with Java Bean objects. The Java Bean object has two groups of characteristics: properties and methods.

Properties are descriptive features of the Java Bean object. Such features preferably include the OMS (Object Mapping Structure) which is the text, structure, graphics and APPS intelligence of the Java Bean object. The latter feature, APPS intelligence, is applicable only if the original document was a paper document scanned into an electronic file, since APPS stands for "Adaptive Probability Pattern Search", which enables text to be searched in an image even if not correctly recognized by the OCR process described previously. The OMS contains information related to the overall structure of the Java Bean object, as well as a description of the relationships between different portions of that object.

Preferably, the profile information is also included. The profile information includes any additional desired characteristics of the original document. These characteristics are determined by the user through IC Content Capturer **52**. For example, the profile information could include data concerning the type of company which published the original document Thus, the profile information is external to the original document and is added according to the specification of IC Content Capturer **52**.

Other preferred properties include an optional but preferable object image, which is a visual image of the original document. Another preferred property is hyperlink information, which describes all connections to locations on the World Wide Web. Preferably, a description of the relationships between this component and other components is also provided. Finally preferably security and access control data is provided, which determines who is allowed to access the information.

Methods determine the ways in which the data and properties of the information component can be manipulated. These methods are standard for the Java Bean component architecture. For example, methods include ways to access the data, whether as an image, a video clip, a sound and so forth. Methods also include an application interface, so that another application would be able to interact with the information component and with the stored data, and with a GUI (graphical user interface). Other methods pertain to access control and to event handling. Event handling, as noted previously in section 1, is the mechanism for Java Bean components to broadcast events and to have those events delivered to an appropriate component or components for notification. Thus, event handling provides methods for communication between components packaged as Java Beans.

Although individual methods might be specific to a particular information component, such that a newspaper article component would probably not include a method for manipulating sound, the overall mechanism for describing each method is supported by

the Java Bean component architecture and could be easily determined by one of ordinary skill in the art.

The information component is preferably packaged as a Java Bean by using the JAR file format. The JAR file format includes such information as the class file, images, sounds and links to other components. The class file is a description of the information class to which the information component belongs. Each such piece of information is stored in the JAR file format as a pointer to the storage location to the relevant data, such as an image for example. Thus, the JAR file format wraps additional information and data around the information component, in such a way that all of the information and data is both presented as a single, independent entity, yet is readily accessible to other software objects.

Section 2: Specific Implementation of the Server with Java Bean Objects in a CORBA Environment

This section describes a specific implementation of IC Server **62**, described in Chapter II, Section 6, as IC JBC Server **300** for operation with Java Bean objects in a CORBA environment.

In this particular embodiment of the present invention, when a client application issues a request for information, IC JBC Server **300** locates the original information entity, isolates the corresponding information component according to a pointer stored in the Java Bean component for example, and then creates an "object image clip". The object image clip is then sent back to the client application as an HTML file. These functions are described in greater detail with regard to Figure 12.

In Figure 12, IC JBC Server **300** includes a database **302**. Database **302** is both accessible to, and is managed by, an IC Manager **304**. IC Manager **304** is responsible for supplying the main CORBA services, as described in Section 1. Preferably, IC Manager **304** provides these services by being adapted to the main proprietary ORB models which are available, such as the "Cartridge" model of Oracle Corp. (California, USA) or the "Blade" model of Informix™. An ORB is an Object Request Broker, preferably a WRB, or ORB for the "Cartridge" model which is able to communicate with individual cartridges.

The main CORBA services include database and indexing services for search and retrieval engines, and for push applications; database navigation services, distributed viewing, imaging and printing services for the information components; network control and retrieval services; and distributed storage services for information components.

For example, if IC Manager **304** is adapted to the "Cartridge" model, then components are accessed from database **302** through one of a number of cartridges, shown as at least one cartridge **306**. Each cartridge **306** is a module of software which performs a specific function. Each of the previously described services performed by IC Manager **304** is provided by a separate cartridge **306**. Different cartridges **306** could provide database indexing, database navigation and information component retrieval services for example, without requiring cartridge **306** and database **302** to be on the same server computer. It should be noted that IC JBC Server **300** is not necessarily a single server computer, but rather is an interacting collection of components which together form IC JBC Server **300**.

Cartridges **306** would communicate with each other and with any databases through an ORB. One advantage of the "Cartridge" model is that communication between different computers could occur through the World Wide Web, via an HTTP daemon as described in section 1. Cartridges **306** are named with a combination of the IP address of the server where cartridge **306** is located and the virtual path to the location of cartridge **306** on that server. Thus, IC Manager **304** would preferably be composed of a number of different cartridges **306**, on one server computer or a plurality of server computers, which preferably interact with each other and any other necessary components, such as databases, through the World Wide Web.

IC JBC Server **300** also includes web application server **3080**, which enables IC Manager **304** to send requests and receive information through the Internet. Together, IC Manager **304** and web application server **308** enable specific information components to be retrieved by first activating a particular cartridge **306** and then performing some action through database **302**. Thus, the name of a desired cartridge **306** can be given to IC Manager **304**, which then locates and activates the desired cartridge, through the Internet if necessary.

Once cartridge **306** has been activated, it performs a specific function, such as retrieving an information component from database **302**, for example. The information component can then be distributed through web application server **308**. Of course, any other communication method which enables IC Manager **304** to interact with web application server **308**, and to give the component to web application server **308**, could also be used.

Once the desired original information has been retrieved, it is sent to one of a plurality of image processors **310**. Each image processor **310** transforms the original information, such as a document, into a Searchable Image Format (SIF) file. Each SIF file can be searched, transformed into HyperText and manipulated with copy and paste functions. Each information format preferably has its own image processor **310**, so that for example a

first image processor **310** could manipulate text editor documents, while a second image processor **310** might handle graphics files such as TIF (Tagged Image Format) or GIF (Graphics Interchange Format) format files, for example. Furthermore, each image processor **310** is optionally and preferably able to transform the "original" information into the corresponding SIF file "on the fly". Thus, the SIF file can be created and recreated as needed, without the requirement of storing both the SIF file and the "original" information.

One example of how such a SIF file could be created from a paper document is given in U.S. Application No. 08/625,496, herein incorporated by reference in its entirety, it being understood that this is only for illustrative purposes only and is not meant to be limiting. SIF files are preferably actually image files, most preferably fully compatible with the TIF file format, which incorporate both graphic images and information data stored in a separate text file, as well as the structure which relates the graphic and textual information within the original document.

SIF files include a header section for general information about the file such as the image resolution, the digital graphic image stored in the conventional raster format, information relating to individual words or elements of the image file, and administrative information which contains the relational structure of the image and textual elements. The information relating to individual words includes not only the text of the words, but also the data generated by the OCR technology regarding unidentified characters and probable errors (APPS), if the original document was an electronically scanned paper document. Thus, any search of the textual information can compensate for these unidentified characters and errors.

The actual SIF file is assembled from the basic document elements which were described in Sections 2 and 5. The SIF file is assembled "on the fly" by image processor **310**, and can then be distributed to a client through web application server **308**. Thus, the SIF file would include the text and images from an original document, for example.

According to another preferred embodiment, the client application issues a request for information by sending a polygon to IC JBC Server **300**. This polygon would include the geometrical location of the desired information within a document. The polygon could first be obtained as the results of a search through IC Manager **304**, for example. Once obtained, the polygon would enable IC JBC Server **300** to determine exactly which information to package into the object image clip. For example, the client application might only want to retrieve a single table from a newsletter. The appropriate polygon would be sent to IC JBC Server **300**, which would then pass the request to the appropriate image processor **310**. The table would then be sent as an object image clip to the client application. Thus, rather than

storing the original document as a collection of smaller components, the original document would be stored in its entirety but then retrieved as an individual component or components, if desired.

In preferred embodiments of the present invention, IC JBC Server **300** preferably also includes a view server **312** and a print server **314**. Similarly to IC Manager **304**, both view server **312** and print server **314** may provide these services by being adapted to the main proprietary ORB models which are available, such as the "Cartridge" model of Oracle Corp. (California, USA) or the "Blade" model of Informix™. For example, print server **314** preferably allows high quality on-demand printing of the original document in a platform-independent manner. Each separate printing service is provided as a cartridge if the "Cartridge" model is used.

View server **312** provides the appropriate image application services to IC Manager **304**, such as services related to the display of an image on a computer screen through a GUI, for example. View server **312** could also provide each service as a cartridge if the "Cartridge" model is used.

Section 3: Client as a Web Browser

In the example shown in Figure 13, the GUI could be an HTML (hypertext mark-up language) interface, such that IC client **98** is a Web browser-type software application, it being understood that this is for the purposes of illustration only and is not meant to be limiting in any way. A similar HTML rendering engine could be used as that described in Chapter III, Section 4. Preferably, an HTML interface **316** displays the Web page. HTML interface **316** could be a Web browser, for example. In addition, preferably the Web page which is displayed is customizable for a particular user by an HTML customization module **318**. Also preferably, Java components **320** can also be provided to client **98**.

While the invention has been described with respect to a limited number of embodiments, it will be appreciated that many variations, modifications and other applications of the invention may be made.

WHAT IS CLAIMED IS:

1. A information component system for storing an original document, comprising:

(a) at least one information component for storing information from the original document:

(b) an information component identifier for classifying said at least one information component according to at least one information component class; and

(c) at least one property of said at least one information component.

2. The information component system of claim 1, further comprising:

(d) a hierarchy of related classes, such that at least one property of said at least one information component is determined according to a location of said at least one information class in said hierarchy.

3. The information component system of claim 2, wherein said information class is selected from the group consisting of newspaper class and video stream class.

4. The information component system of claim 3, wherein said newspaper class includes at least one information class selected from the group consisting of article, page and picture.

5. The information component system of claim 3, wherein said video stream class includes at least one information class selected from the group consisting of video clip and video frame.

6. The information component system of claim 2, further comprising:

(e) a software system for displaying the original document in an original format; and

(f) a content capturer for capturing said information from the original document by interacting with said software system.

7. The information component system of claim 6, wherein the original document is selected from the group consisting of an XML document, a word processing document, a PDF file, a video stream, an audio stream and a Web page.

8. The information component system of claim 7, wherein said software system is selected from the group consisting of word processor, facsimile machine software, Web browser and Adobe™ Acrobat™.

9. The information component system of claim 8, wherein said software system is said word processor, said word processor featuring a printer driver for printing, and said content capturer captures said information through said printer driver of said word processor.

10. The information component system of claim 8, wherein said content capturer captures said information through a clipboard.

11. The information component system of claim 6, further comprising:

(g) an information component identifier for identifying said at least one information component class of said at least one information component.

12. The information component system of claim 11, further comprising:

(h) a rules table for containing at least one rule according to which said at least one information component class is determined; and

(i) a knowledge base for storing said at least one information component class and said rules table.

12. The information component system of claim 11, wherein said at least one rule features a plurality of tokens, each of said plurality of tokens determining a property of said information component.

13. The information component system of claim 12, wherein said property of said information component is a geometrical location of said information component within the original document.

14. The information component system of claim 13, wherein said property of said at least one information component is a textual attribute.

15. The information component system of claim 11, further comprising:

(h) a contributor for transforming said at least one information component into a software object.

16. The information component system of claim 15, wherein said software object is an XML-environment compatible object.

17. The information component system of claim 16, wherein said XML-environment compatible object features:

(i) data obtained from said at least one information component;

(ii) at least one method for accessing said data; and

(iii) at least one property of said XML-environment compatible object.

18. The information component system of claim 17, wherein said at least one property is a location of said at least one information component in said information component hierarchy.

19. The information component system of claim 18, wherein a link between said XML-environment compatible object and at least one other XML-environment compatible object is determined according to said location.

20. The information component system of claim 19, wherein said link is stored as a link object, said link object being a child of said XML-environment compatible object.

21. The information component system of claim 15, wherein said software object is a CORBA-compatible object.

22. The information component system of claim 15, wherein said software object is selected from the group consisting of a COM object, a Java Bean component having data stored according to the JAR format and a flat file.

23. The information component system of claim 15, further comprising:

(i) a database for storing said at least one information component; and

(j) a server for placing said at least one information component in said database and for retrieving said at least one information component upon receiving a request.

24. The information component system of claim 23, wherein said server is an XML server and said at least one information component is an XML-environment compatible object.

25. The information component system of claim 24, wherein said XML server assembles an XML document upon receiving said request by retrieving said XML-environment compatible object and by placing said XML-environment compatible object into said XML document.

26. The information component system of claim 25, wherein said XML-environment compatible object is linked to at least one other XML-environment compatible object through a link object, said link object being a child of said XML-environment compatible object.

27. The information component system of claim 26, wherein said link object is identified in a link table, and said XML-environment compatible object is identified in an IC table, said link table and said IC table being stored in said database, and said XML server manages said link table and said IC table, such that said link table is altered when said IC table is altered.

28. The information component system of claim 25, wherein said XML document is assembled according to a style sheet, said style sheet featuring at least one rule for displaying said XML-environment compatible object as an XML element.

29. The information component system of claim 25, further comprising:

(k) an XML editor for editing an XML document; and

(l) a DOM interface for interfacing said XML editor and said XML server.

30. The information component system of claim 25, further comprising:

(k) a normalizer for normalizing a tag of said XML element according to at least one rule.

31. The information component system of claim 23, further comprising:

(k) a manager for placing said at least one information component into said database, and for retrieving said at least one information component from said database.

32. The information component system of claim 31, wherein said manager is an object request broker (ORB), such that said request is performed as an object request.

33. The information component system of claim 32, wherein said server further comprises:

(i) at least one Cartridge for performing a service with said at least one information component through said manager.

34. The information component system of claim 23, further comprising:

(k) a publisher for publishing said at least one information component, said publisher receiving said at least one information component from said server; and

(l) a client for receiving said at least one information component from said publisher and for displaying said at least one information component.

35. The information component system of claim 34, wherein said client is a Web browser software application.

36. The information component system of claim 35, wherein said publisher further comprises an HTML rendering engine for rendering said at least one information component as an HTML document.

37. The information component system of claim 1, wherein said at least one information component is a CORBA-compatible object.

38. The information component system of claim 1, wherein said at least one information component is selected from the group consisting of a COM object, a Java Bean component having data stored according to the JAR format and a flat file.

39. The information component system of claim 1, wherein said at least one information component is an XML-environment compatible object.

40. The information component system of claim 1, wherein said information from the original document includes textual data, image data and visual attributes.

41. The information component system of claim 40, wherein said visual attributes are selected from the group consisting of font type, font style and location of said textual data.

42. The information component system of claim 1, wherein said at least one information component is a higher information component, and said at least one information component is further divided into a plurality of lower information components, such that each of said plurality of lower information components is capable of being shared between a plurality of said higher information components.

43. A system for displaying a native file format document, the document including text and having a native file format and a native document appearance, the native file format including at least one instruction for displaying the text of the native file format document, the system comprising:

(a) a Web browser for displaying the native file format document according to the native document appearance; and

(b) a HTML rendering engine for obtaining information regarding the native document appearance of the native file format document, for translating said information into a raster file having a raster format displayable by said Web browser, and for giving said translated information to said Web browser, such that said Web browser is able to display the native file format document.

44. The system of claim 43, further comprising an indexing database, said indexing database containing a plurality of records, each record having a word of the text

from the native file format document, such that the text is searchable according to a request from said Web browser.

45. The system of claim 44, wherein each of said records further includes coordinates of said word.

46. The system of claim 44, wherein said coordinates include at least one feature selected from the group consisting of a number designating a page from the native file format document, a number of said word on said page and coordinates of a bounding rectangle of said word on said page.

47. The system of claim 46, wherein each of said records further includes a visual attribute of said word.

48. The system of claim 6, wherein said visual attributes include at least one feature selected from the group consisting of font size, font style, character attribute and space between characters of said word.

49. The system of claim 45, wherein said HTML rendering engine is able to search said indexing database according to said request from said Web browser and to give a result to said Web browser, such that said result is displayable by said Web browser according to the native document appearance.

50. The system of claim 49, wherein said result includes a plurality of matches between said request and the native file format document, such that said HTML rendering engine is able to select one of said plurality of matches according to an instruction from said Web browser.

51. The system of claim 43, further comprising:

(c) a native file format processor for accessing the native file format document through the native file format, such that the at least one instruction of the native file format enables said native file format processor to display the native file format document according to the native document appearance, and

such that said HTML rendering engine receives said information about the native file format document from said native file format processor.

52. The system of claim 51, wherein said HTML rendering engine receives said information from said native file format processor through a printer driver.

53. The system of claim 43, wherein said HTML rendering engine obtains said information by substantially directly accessing the native file format document.

54. A method for managing information, comprising the steps of:

(a) capturing the information in an electronic format;

(b) converting said captured information into an information component, said information component featuring:

(i) a pointer to a storage location of said captured information;

(ii) at least one method for manipulating said captured information; and

(iii) at least one property of said captured information;

(c) storing said information component; and

(d) displaying said information component such that said captured information appears in substantially the original format.

55. A information component comprising a software object, said software object including:

(a) a pointer to a storage location of said stored original information;

at least one method for manipulating said stored original information; and

(c) at least one property of said stored original information;

56. The information component of claim 55, wherein said software object belongs to an information class, said information class belonging to a hierarchy of related classes, such that said information class has a pool of attributes according to a location in said hierarchy.

57. The information component of claim 56, wherein said hierarchy includes said information class and at least one information sub-class, such that said pool of attributes of said information class is inherited by said at least one information sub-class.

58. The information component of claim 57, wherein said information class is selected from the group consisting of newspaper class and video stream class.

59. The information component of claim 58, wherein said newspaper class includes at least one information subclass selected from the group consisting of article, page and picture.

60. The information component of claim 58, wherein said video stream class includes at least one information subclass selected from the group consisting of video clip and video frame.

61. A server for serving stored information to a client Web browser, said server comprising:

(a) a database for storing the stored information; and

(b) an image processor for accessing the stored information from said database and transforming the stored information into a Searchable Image Format (SIF) file, said SIF file being accessed by the client Web browser, such that the stored information is displayed by the client Web browser.

62. The server of claim 61, wherein said SIF file includes:

(i) a raster image of the stored information;

(ii) a text of the stored information: and

(iii) a relationship between said text and said raster image, such that a location of said text within said raster image is specified.

63. The server of claim 61, further comprising a polygon sent from the client Web browser to the server, said polygon specifying a portion of the stored information to be sent to the client Web browser, such that said SIF file includes a raster image of said portion of the stored information and such that substantially only said portion of the stored information is displayed by the client Web browser.

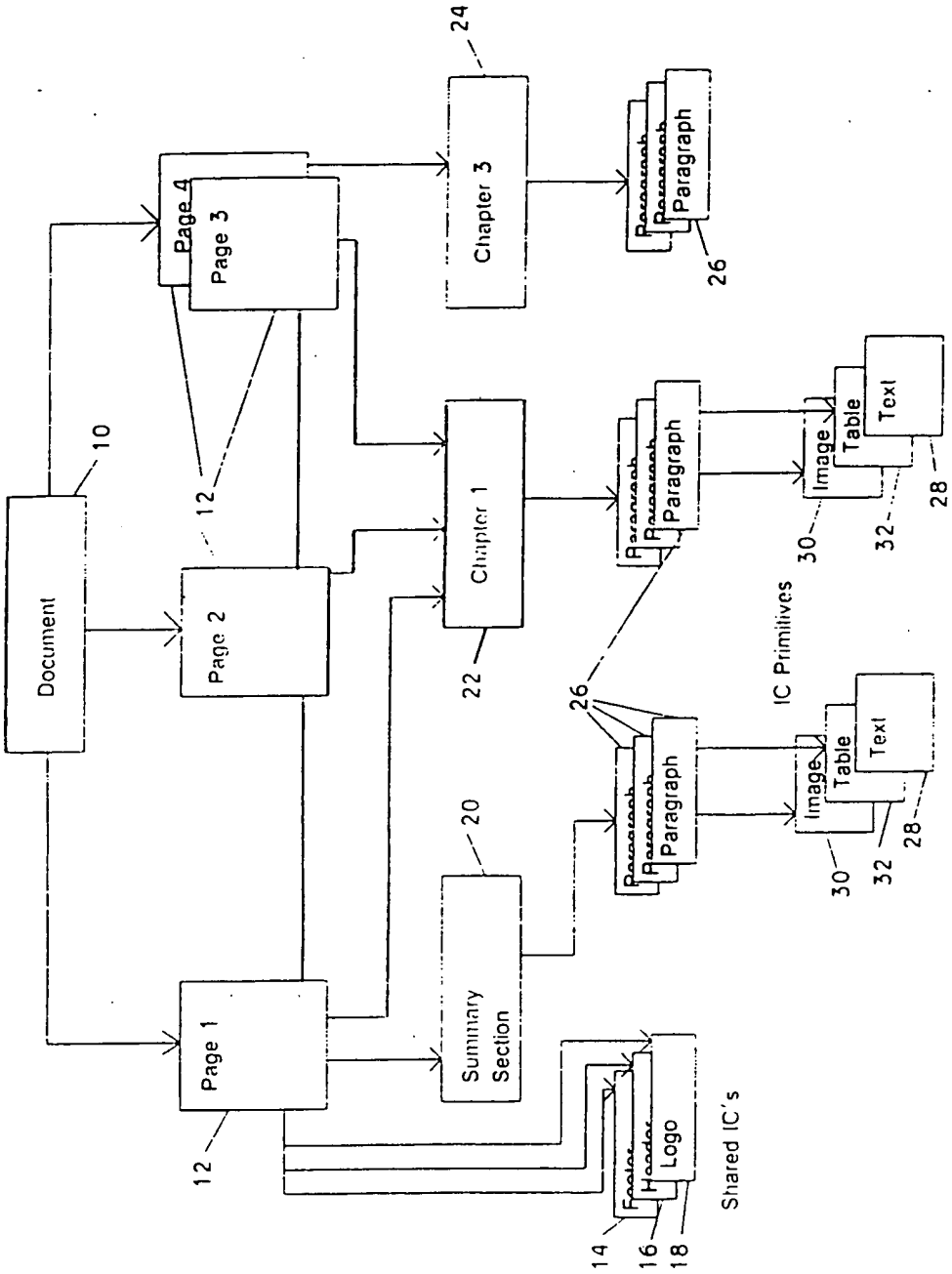

Document
10
Page 1
Page 2
Page 3
Page 4
12
Chapter 1
Chapter 3
22
24
Summary Section
20
Paragraph
26
IC Primitives
Image
Table
Text
28
30
32
Logo
Header
Footer
Shared IC's
14
16
18

FIG. 1A

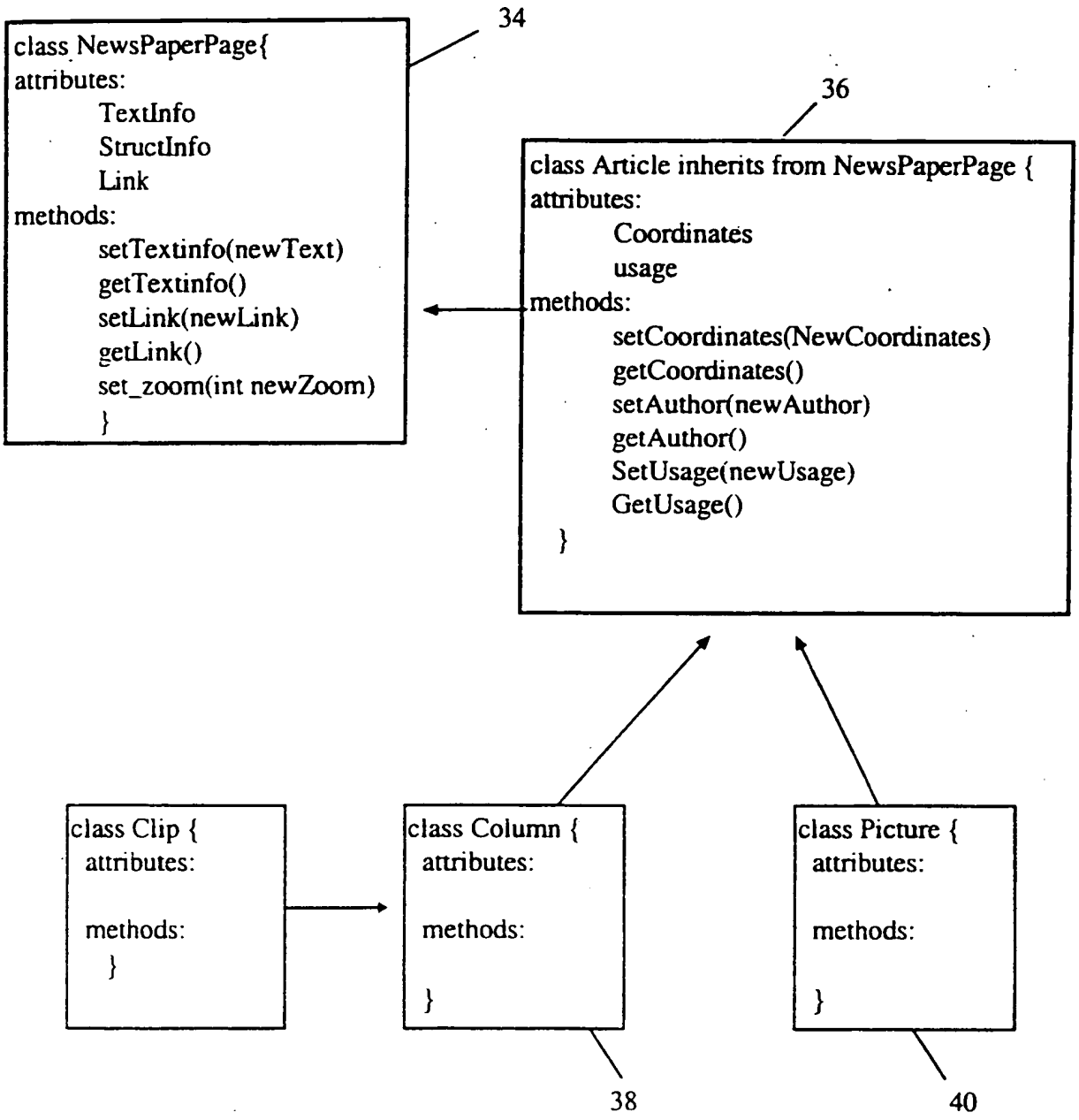

34
class NewsPaperPage{
attributes:
TextInfo
StructInfo
Link
methods:
setTextinfo(newText)
getTextinfo()
setLink(newLink)
getLink()
set_zoom(int newZoom)
}
36
class Article inherits from NewsPaperPage {
attributes:
Coordinates
usage
methods:
setCoordinates(NewCoordinates)
getCoordinates()
setAuthor(newAuthor)
getAuthor()
SetUsage(newUsage)
GetUsage()
}
class Clip {
attributes:
methods:
}
class Column {
attributes:
methods:
}
class Picture {
attributes:
methods:
}
38
40

Figure 1B

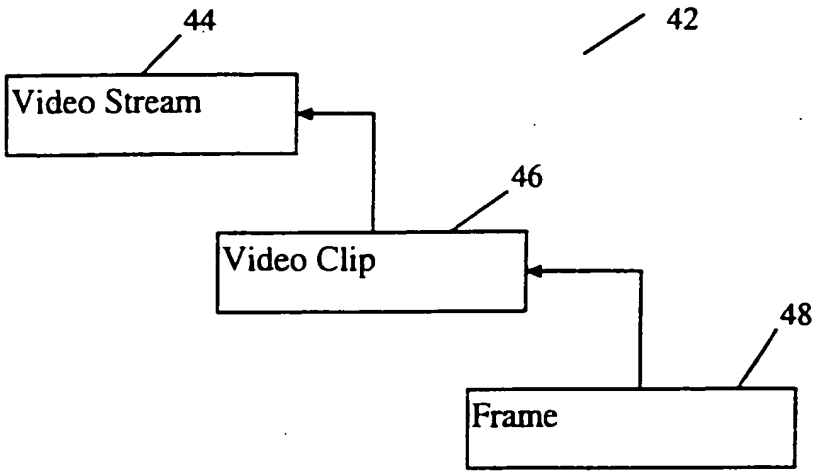



Figure 1C

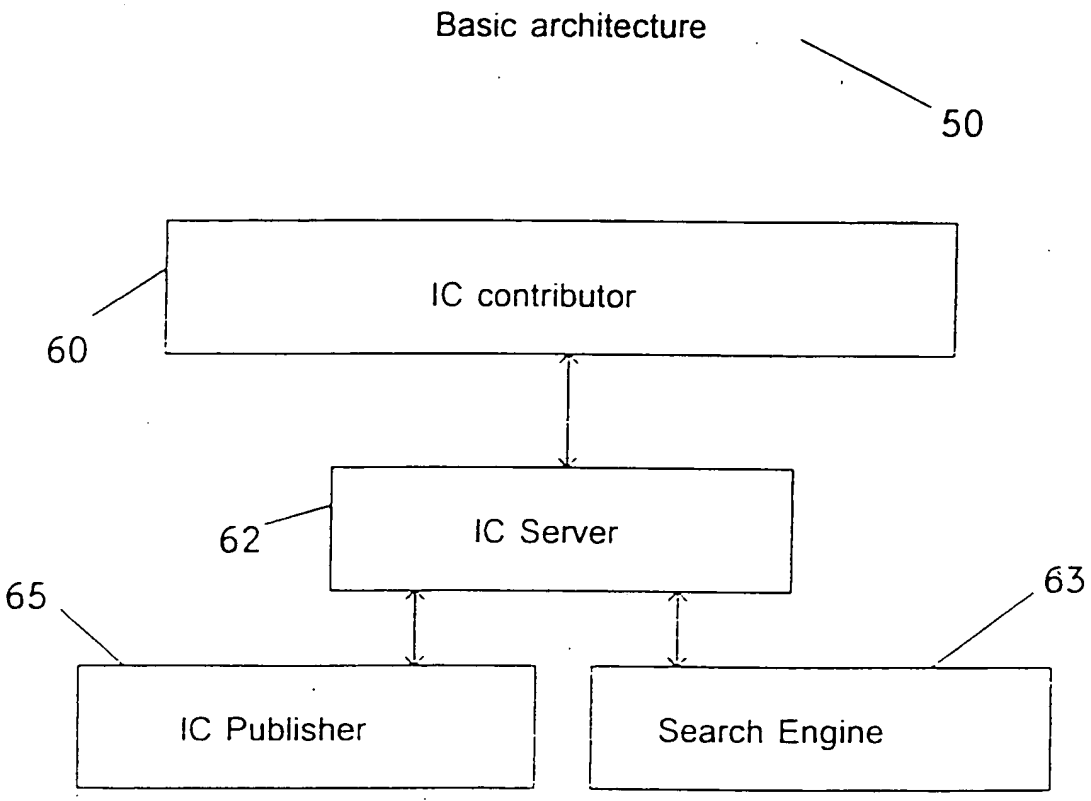

Basic architecture
50
IC contributor
60
IC Server
62
65
63
IC Publisher
Search Engine

FIG. 2A

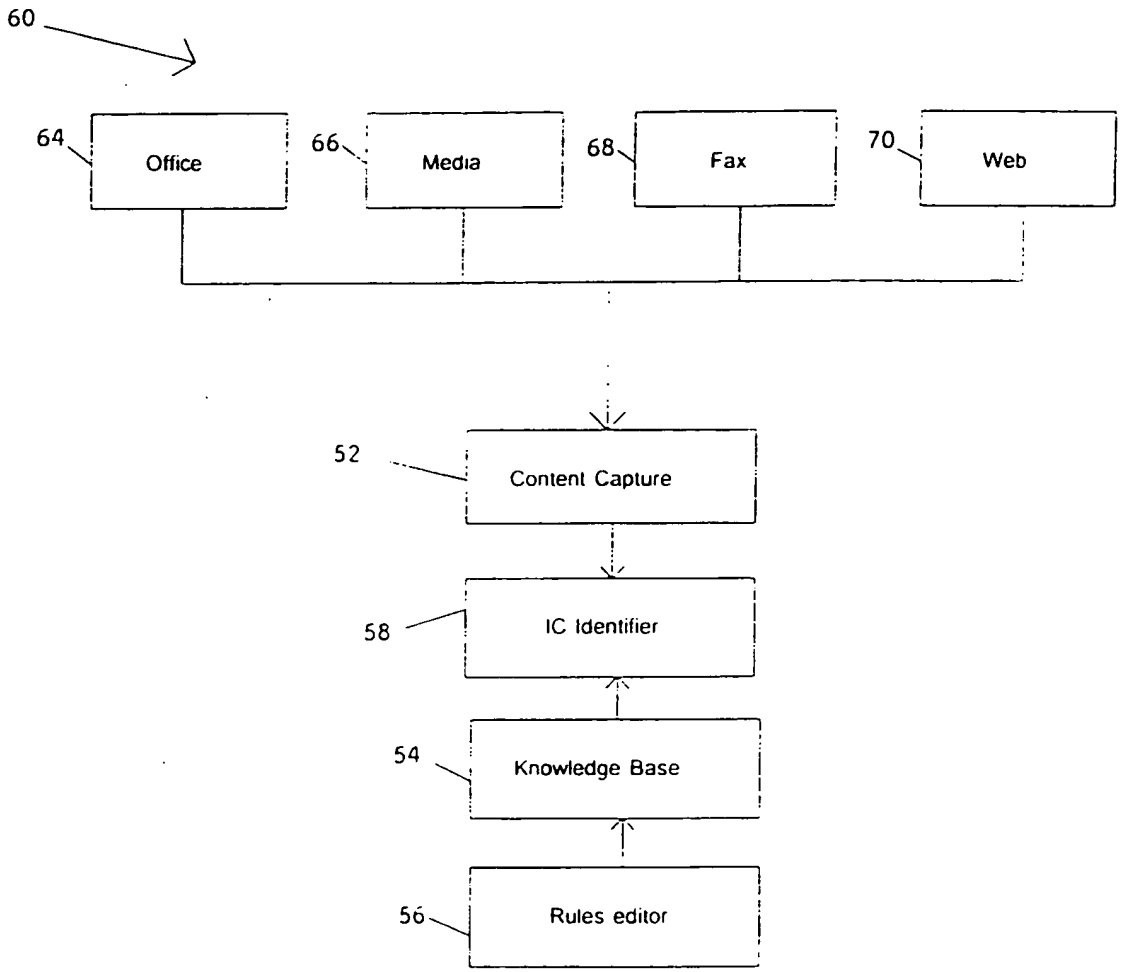
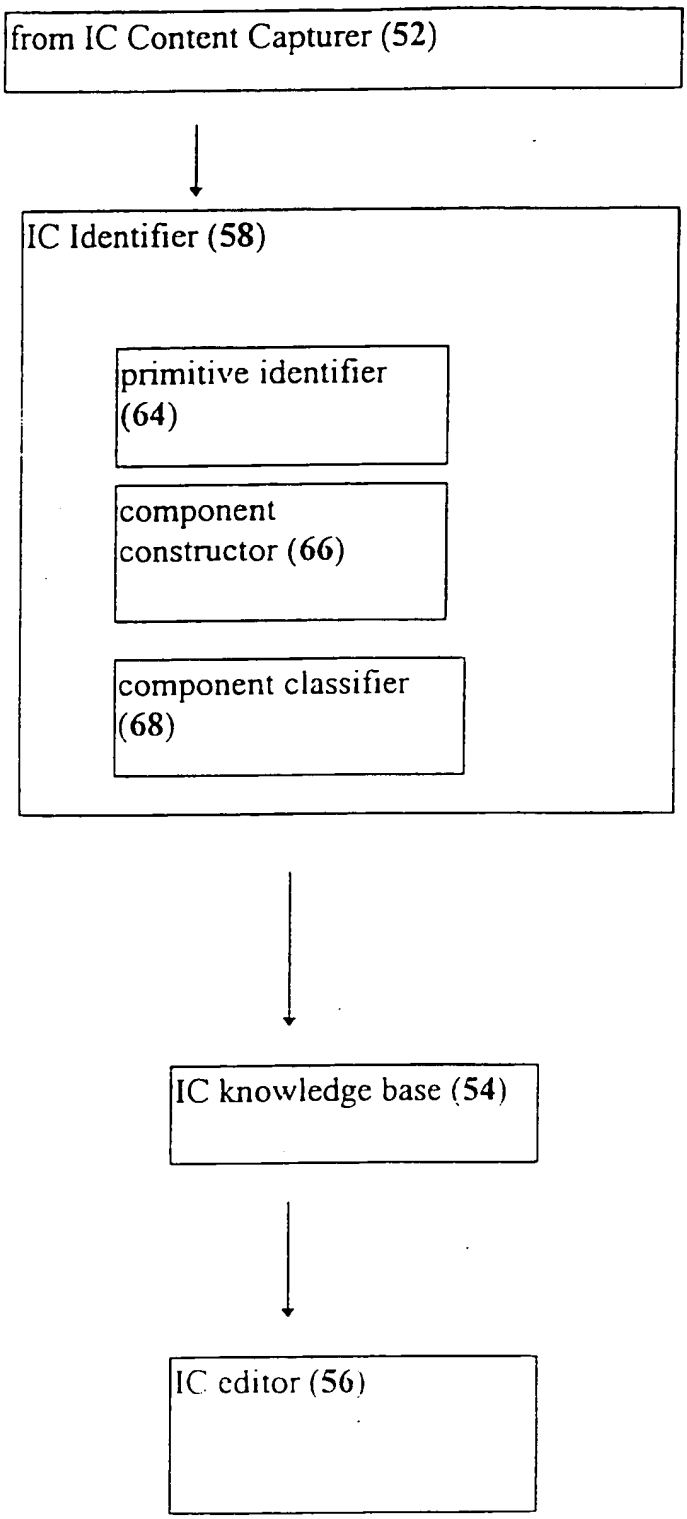

60
64
Office
66
Media
68
Fax
70
Web
52
Content Capture
58
IC Identifier
54
Knowledge Base
56
Rules editor

FIG. 2B

Figure 3A


from IC Content Capturer (52)
IC Identifier (58)
primitive identifier (64)
component constructor (66)
component classifier (68)
IC knowledge base (54)
IC editor (56)

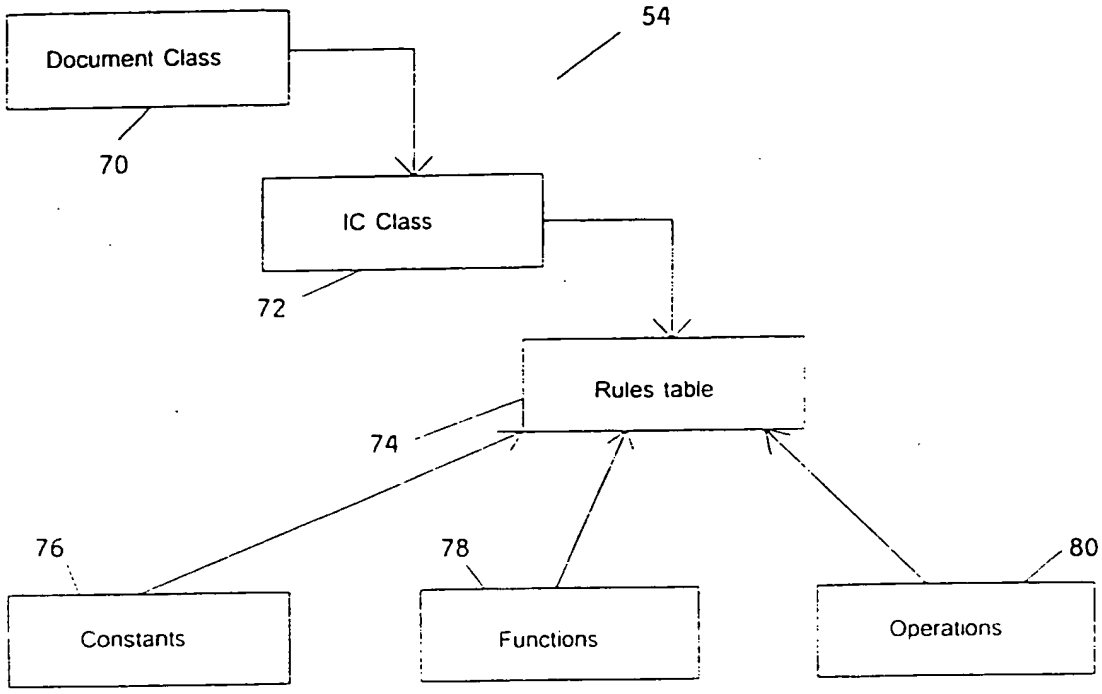

Document Class
54
70
IC Class
72
Rules table
74
76
78
80
Constants
Functions
Operations

FIG. 3B

Figure 4

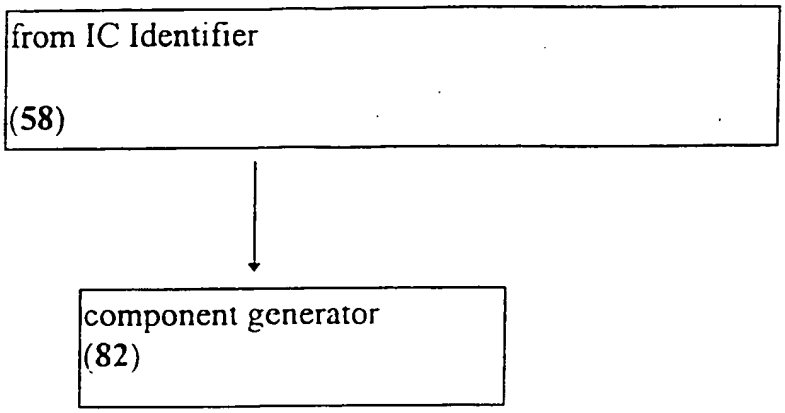


Figure 5

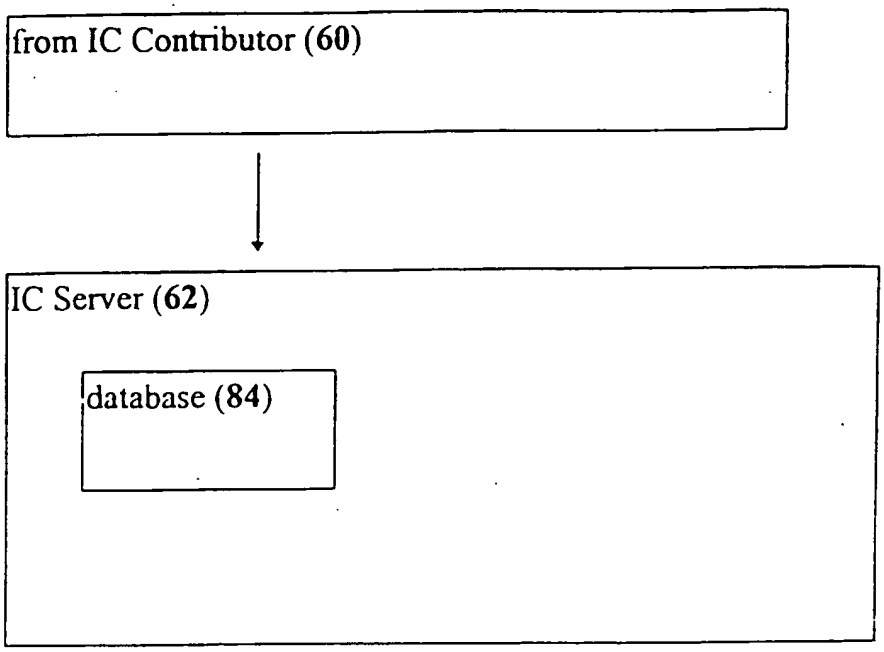
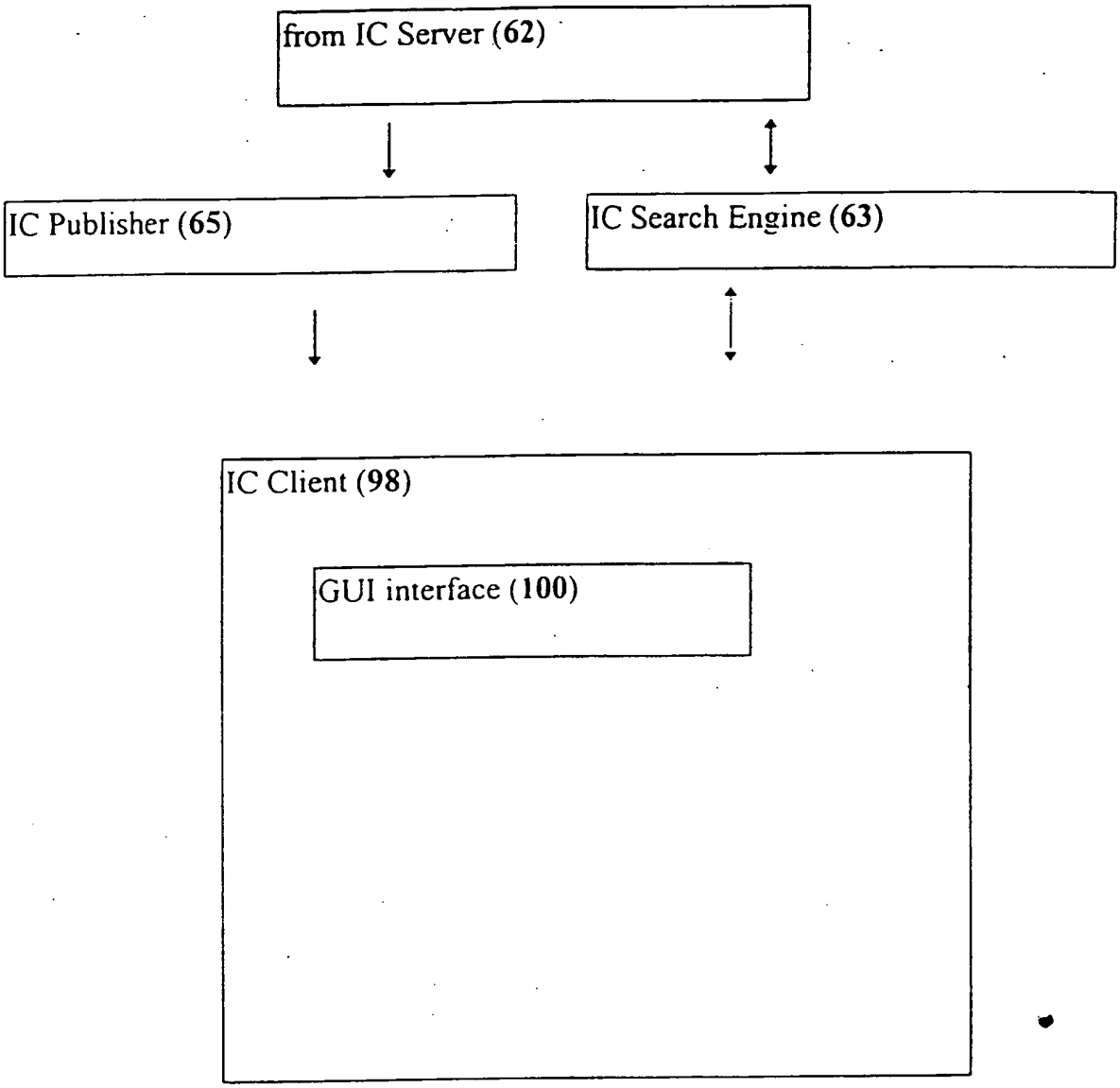


Figure 6


from IC Server (62)
IC Publisher (65)
IC Search Engine (63)
IC Client (98)
GUI interface (100)

Basic architecture

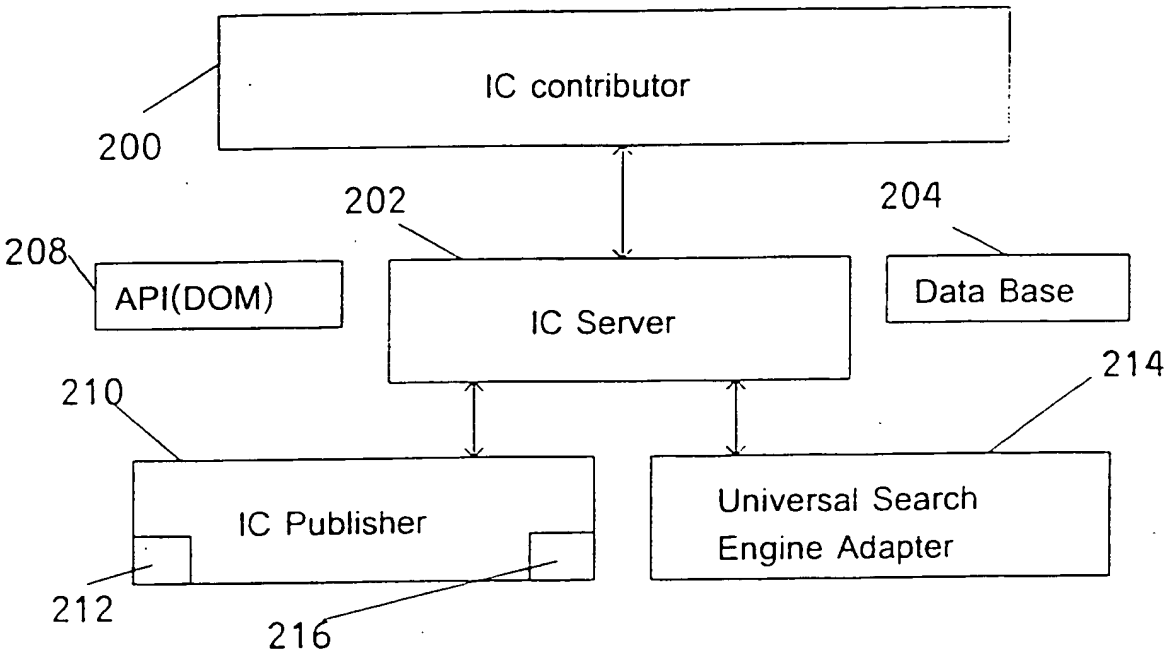



FIG. 7

# Dynamic Link Management

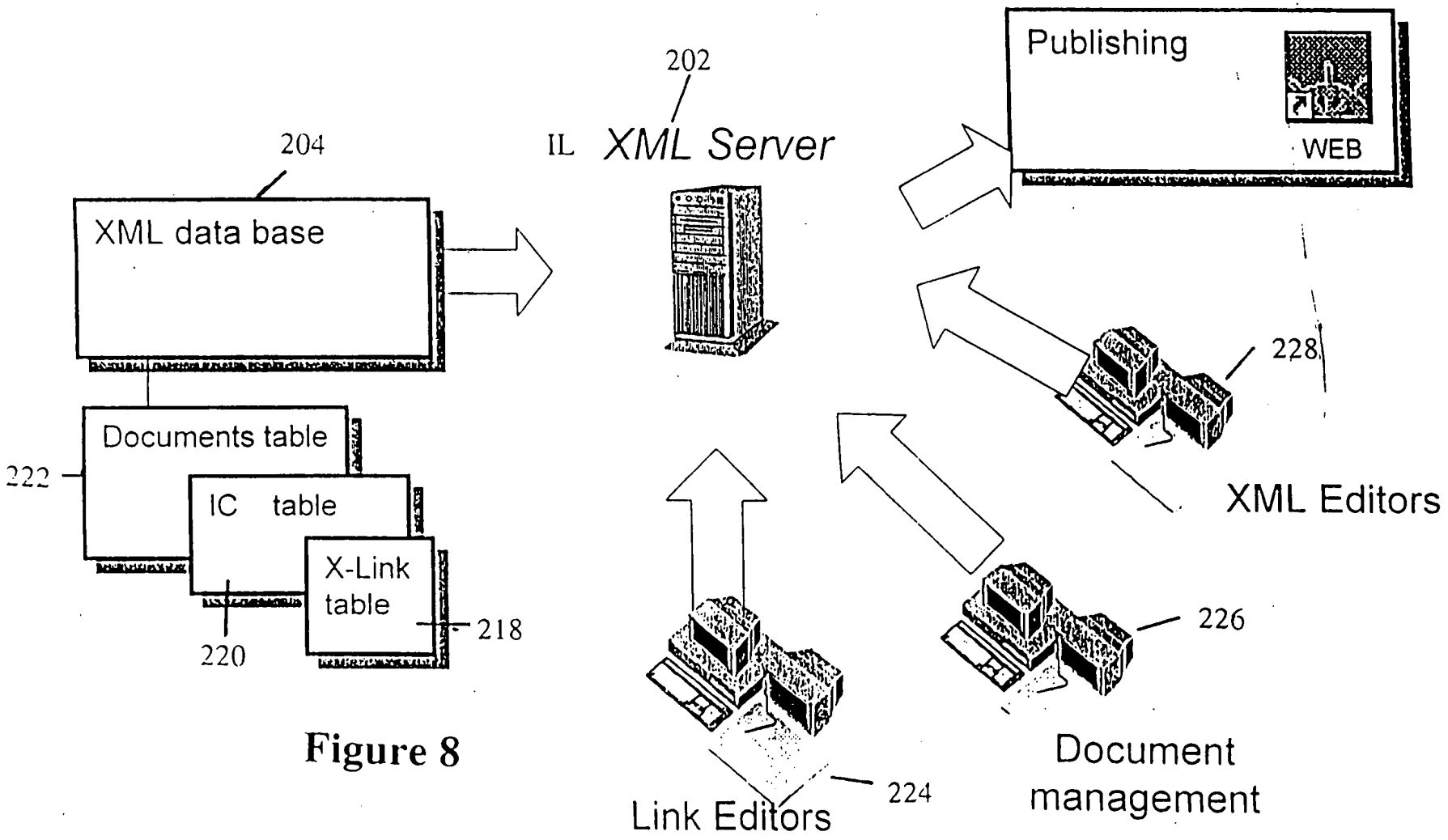



Figure 8

DTD Normalization process

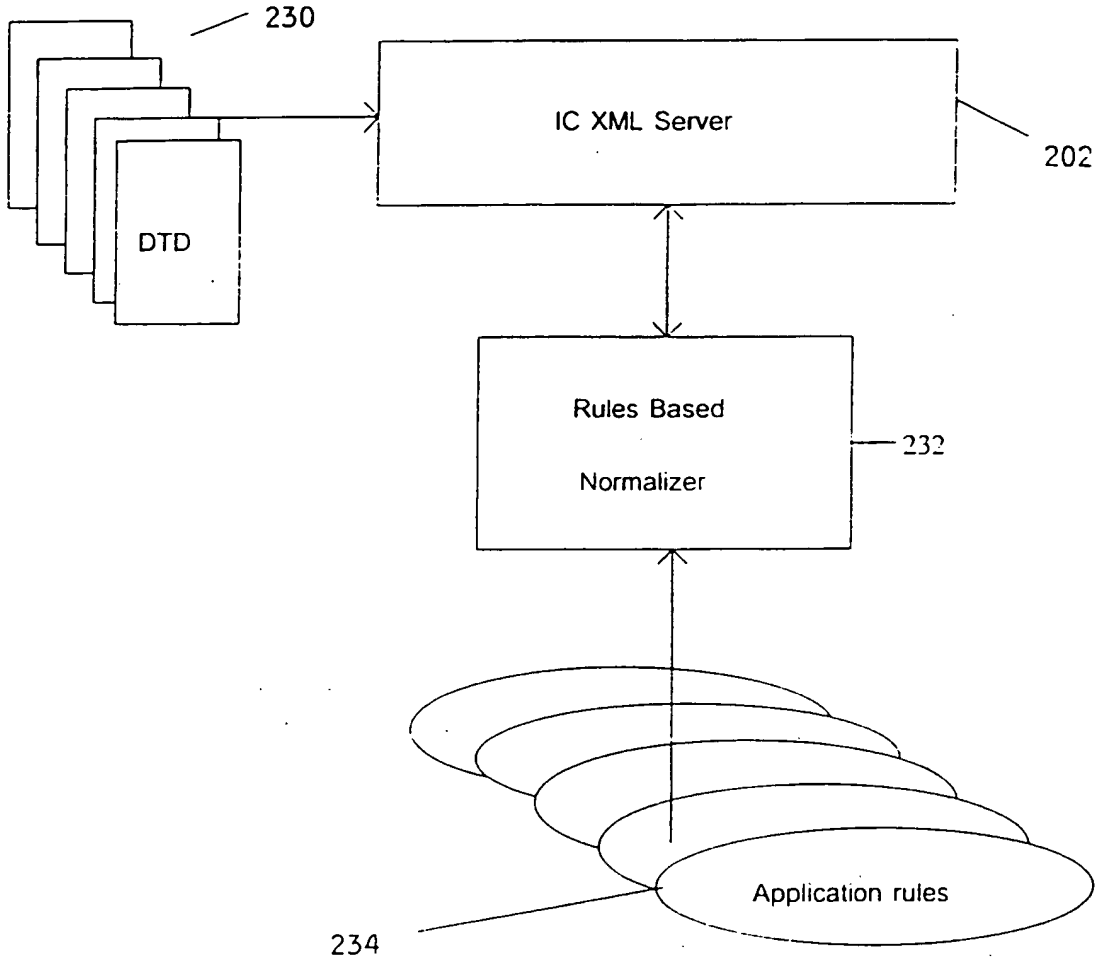



FIGURE 9

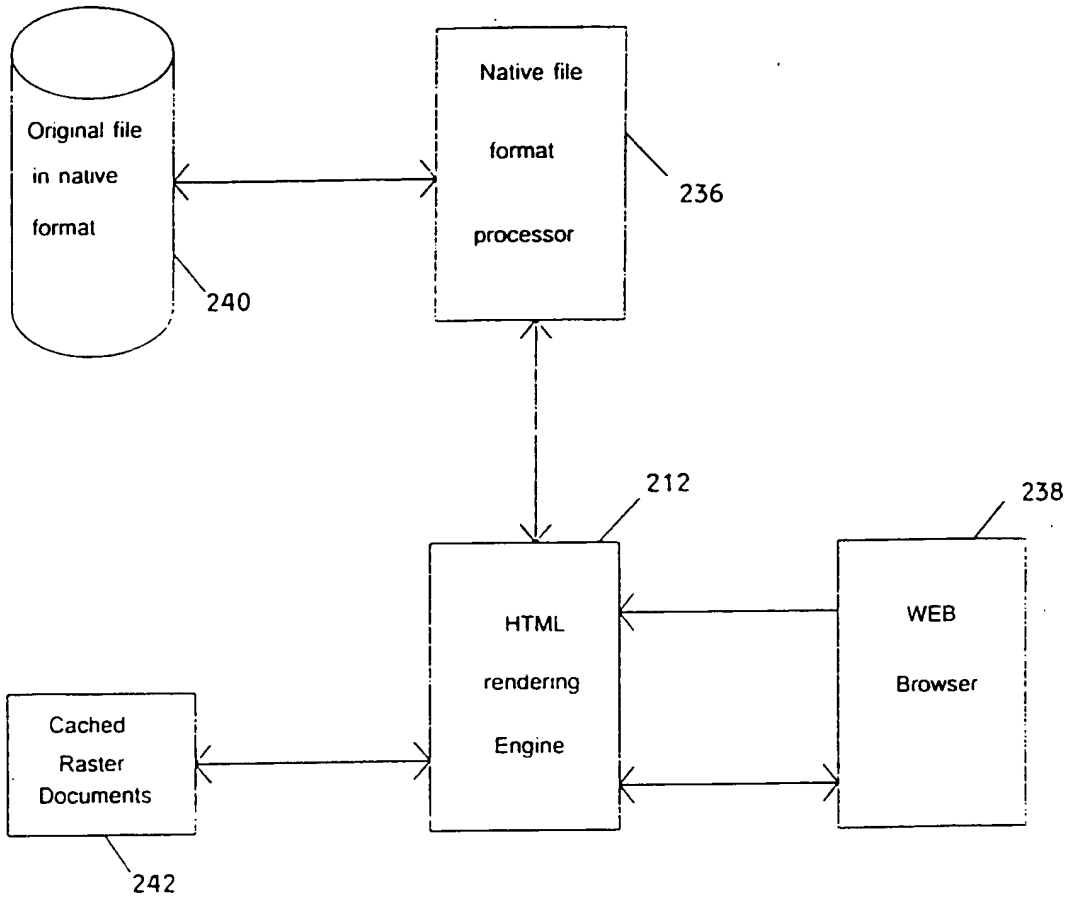

Original file in native format
240
Native file format processor
236
212
238
HTML rendering Engine
WEB Browser
Cached Raster Documents
242

FIG. 10

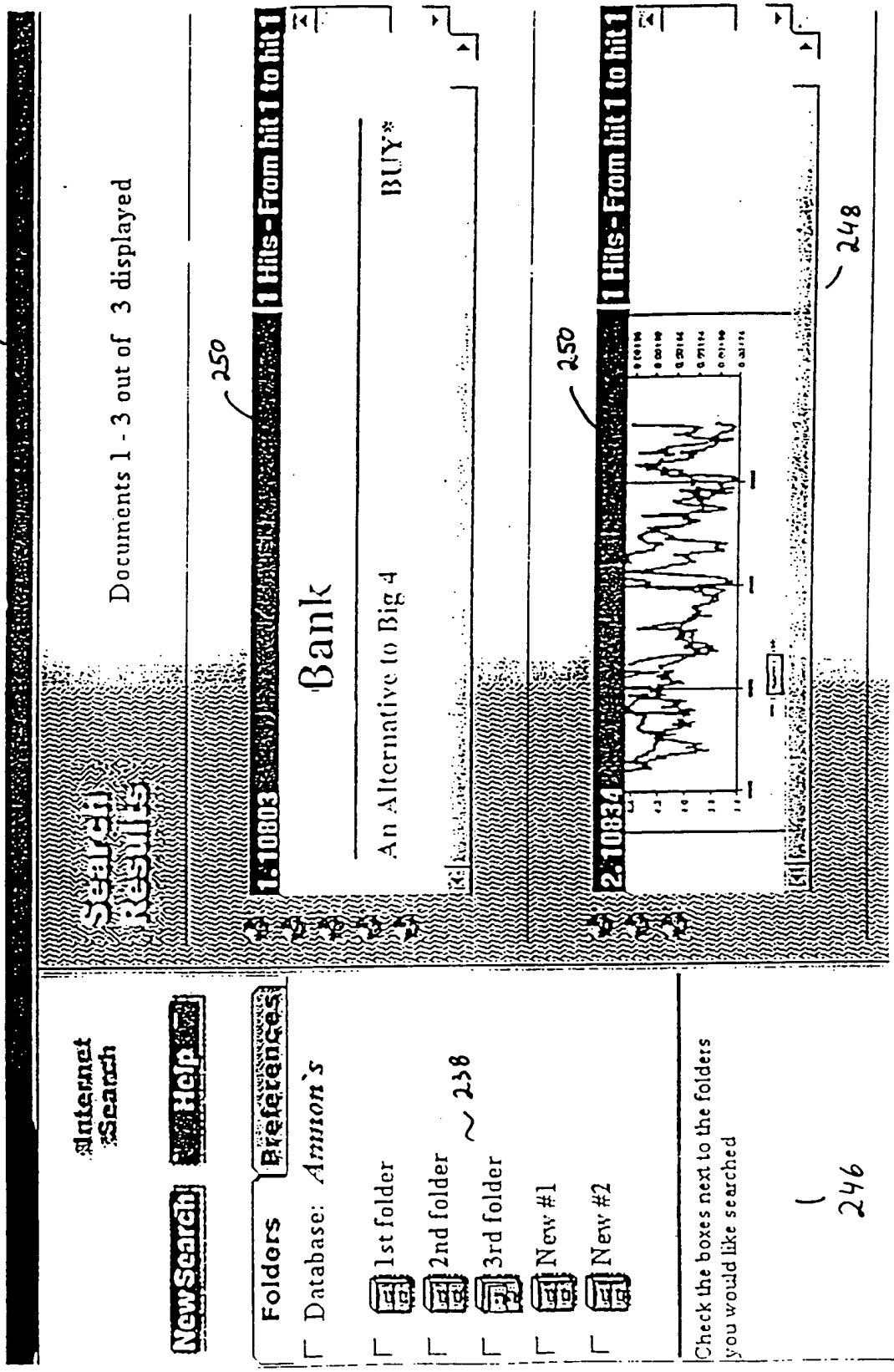

244
Internet Search
New Search
Help
Search Results
Documents 1 - 3 out of 3 displayed
Folders
Preferences
Database: Amion's
1st folder
2nd folder
3rd folder
New #1
New #2
238
Check the boxes next to the folders you would like searched
246
250
1 Hits - From hit 1 to hit 1
1. 10803
Bank
An Alternative to Big 4
BUY*
2. 10834
248

FIG. 11

Figure 12

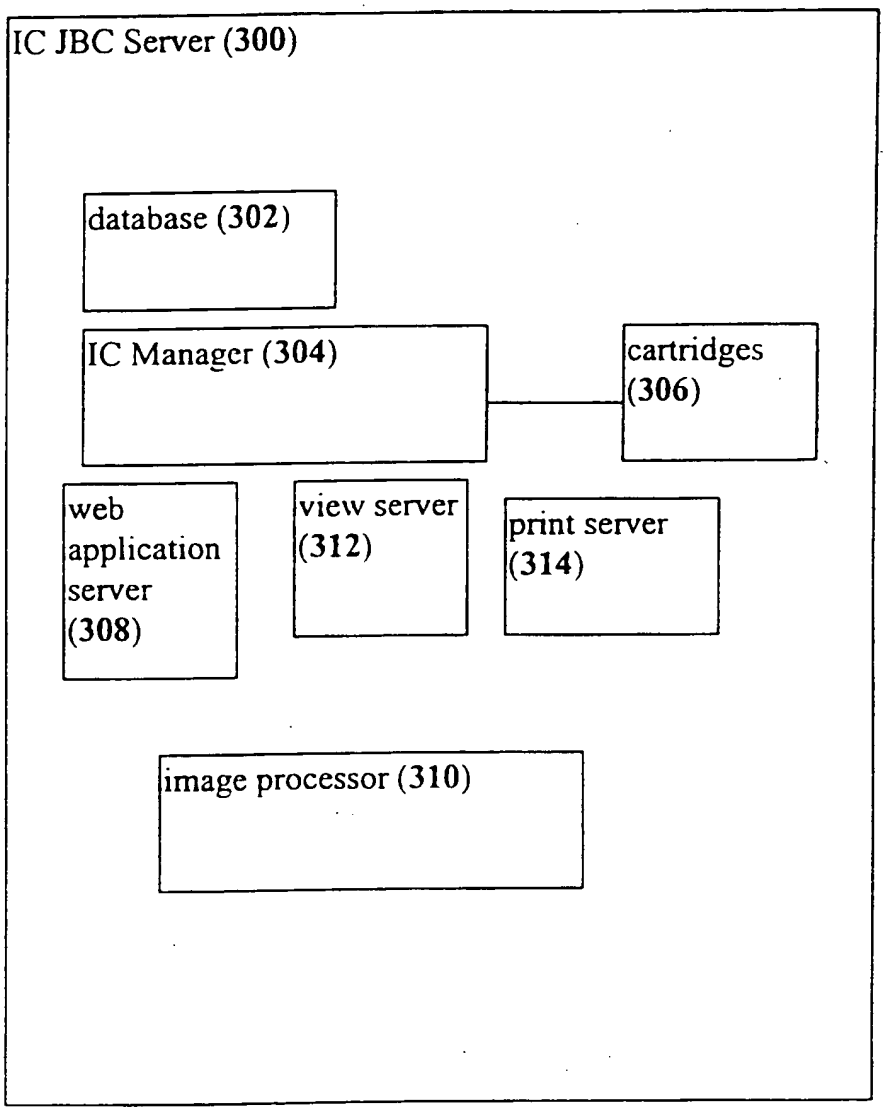
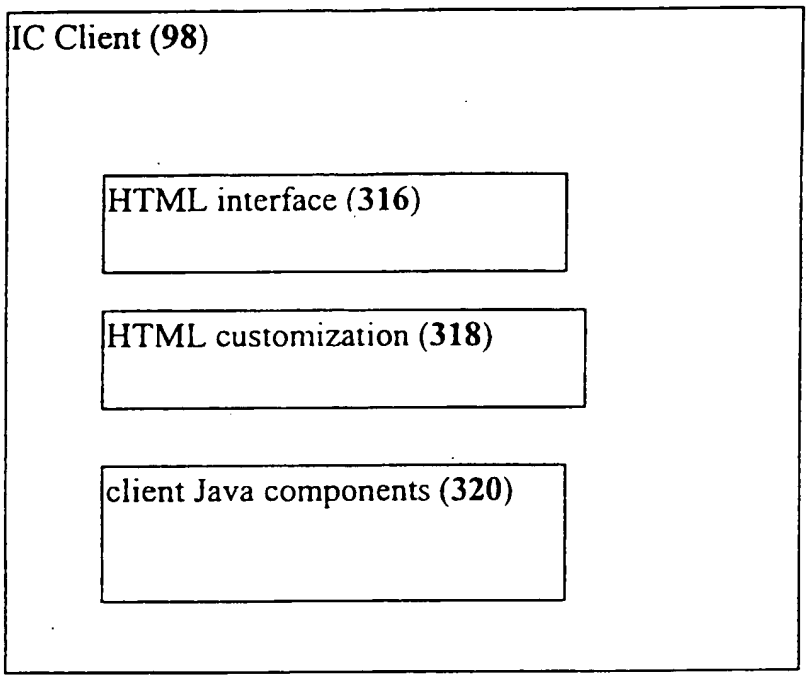

IC JBC Server (300)
database (302)
IC Manager (304)
cartridges (306)
web application server (308)
view server (312)
print server (314)
image processor (310)

Figure 13


IC Client (98)
HTML interface (316)
HTML customization (318)
client Java components (320)